UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Métodos de clustering en datos de expression génica

Autor:
Aurora Torrente Orihuela

Directores:
Dr. Alvis Brazma
Dr. Juan Romo

DEPARTAMENTO DE ESTADISTICA

Leganés, marzo de 2007

A papá y mamá

Aknowledgements

At last, the time of ending up has arrived. And it seems the most difficult part because compressing in a few words all the gratefulness I feel for all those people that somehow have been by my side, giving me support, is a hard task...

The first people I want to thank are my parents, my sisters and my best friend Tere, for having expected from me much more than I ever did, and for being sure all the time that this end would come.

Next, I want to thank Víctor Hernández for introducing me to one of my supervisors, Juan Romo, for giving me the opportunity to start this work and for believing and supporting my work at any time. I am also very grateful to Juan, who gave me always the optimism I often lacked of and for his advices.

I want to specially mention all the colleagues, mates and friends that I have been so lucky to meet along these years: at the department of Statistics in Leganés, thanks to María, Isaac, Angel, Juan Carlos, Conchi, Júlia and Ismael, and in Getafe, to all those people I'm very pleased to have met, who were always willing to help: Charo, Esther, Javier, and of course, Sara, who always was an understanding, supporting friend, and who it's been a pleasure to work with.

Next, at the department of Physics, I want to thank Dani (and Maripaz) because of their friendship and support, and Roberto for his advices. And finally, hundreds of kilometers away,

at my missed EBI, I have to say that I met some incredible people that always were by my side: above all of them I want to thank my other supervisor, Alvis Brazma. I have no words to express my gratitude for his constant help, his brilliant ideas, and for having turned my poor skills and results into pages of work I became proud of. Thank you very much in deed. I also feel grateful to Misha, who always was patient with me and my problems with unix and EP; to Jaak, who always had an interesting point of view of my work; to Gabriella, who helped us with her knowledge about the S.pombe data. And to my friends, Ahmet, who understood and helped me from the very moment we met, Gonzalo, who invariably reduced the running time of my algorithms from O(n) to O(ln n); (without him I would still be running my experiments...), and Marcos, for giving always an optimistic point of view of every problem.

Finally, I want to devote these last lines to the person whose persistence and love made it possible to end this work. My husband, who never let me surrender or step back during these years, who made me start alone the amazing experience of my stay abroad, but finish it by my side, who always tried to make an organized researcher of me (without much success...), who started me up with LaTeX, who always searched for a way out of problems... Too many things to summarize them all here... Each page in this manuscript is devoted to YOU, (and specially to the memories from the Cavendish lab and "my" formula... Those were the days of our lives...). Just thanks...

Índice

Preface

Clustering is an old data analysis problem that has been extensively studied during the last decades. However, there is not a single algorithm that provides a satisfactory result for every data set. Moreover, there exist some problems related to cluster analysis that also remain unsolved. In this monograph we study some of such problems as they commonly appear in practice, and test how they work when applied to gene expression data analysis, where clustering is widely used.

Different clustering algorithms often lead to different results, and in order to make sense out of them it is important to understand how clusters from one analysis relate to those from a different one. A comparison method to find and visualize many-to-many relationships between two clusterings, either two flat clusterings or a flat and a hierarchical clustering, is presented. The similarities between clusters are represented by a weighted bipartite graph, where the nodes are the clusters and an edge weight shows the number of elements in common to the connected nodes. To visualize the relationships between clusterings the number of edge crossings is minimized. When applied to the case of comparing a hierarchical and a flat clustering we use a criterion based either on the graph layout aesthetics or in the mutual information, to decide where to cut the hierarchical tree.

Since iterative methods are sensitive to the initial parameters, we have developed two refinement algorithms designed to improve this initial state, based on the notion of data depth. One of these algorithms looks for initial points in the same data space, while the second one, using the bootstrap technique, selects the initial seeds in a new space of bootstrap centroids.

Also, this second approach allows to construct a soft (non-hard) clustering of the data, that assigns to each point a probability of belonging to each cluster, and thus a single point may partially belong to more than one cluster.

On the other hand, the number of clusters underlying in a data set is usually unknown. Using ideas from the clustering comparison method previously proposed and from the data depth concept, we present three procedures to estimate the number of real groups. The first two methods consist basically in sampling pairs of clusterings from a population and successively performing comparisons between them to find a consensus in the number of clusters, and the third one looks for representative subsets of the clusters whose diameter is used to estimate the optimal number of real groups.

The extensive study we carried out in simulated and real gene expression data shows that the techniques presented here are useful and efficient. The results that we obtained with real data make sense not only from a statistical point of view, but they have proven to have a biological meaning.

Chapter 1

Introduction

Summary

Clustering is an old data analysis problem whose aim is the partitioning of a data set into subsets (clusters) highly internally homogenous (members are similar to one another) and highly externally heterogenous (members are not like members of other clusters). The interest in this technique has been recently revived with the use of new methodologies for the study of gene expression data, such as microarrays. For the understanding of the nature of this type of data, some basics of molecular biology and a short description of how microarrays are built and how gene expression data are produced are needed. We summarize some of the features common to all cluster algorithms, which can be either hierarchical or non-hierarchical (flat): similarity measures, clustering criteria and algorithms themselves. Finally, we present some of the open problems related with clustering analysis that are studied in this work: the comparison of two different clusterings, either hierarchical or flat, the refinement of the initial state of iterative methods and the estimation of the number of clusters.

1.1. The clustering problem

Classification of a set of observations or entities into groups is an old data analysis problem. In Statistics, it is an inherently multivariate problem, whose most common scenario is one in which the data on hand pertain to many variables measured on each entity and not one involving just a single variable. This high-dimensional nature of classification provides an opportunity but also presents some difficulties to the developer of appropriate statistical methodology.

One can distinguish two broad categories of classification problems. In the first, one has data from known or prespecifiable groups as well as observations from entities whose group membership, in terms of the known groups, is unknown initially and has to be determined through the analysis of the data. In the pattern recognition literature this type of classification problem is referred to as supervised learning; in statistical terminology it falls under the heading of discriminant analysis. See Ripley and Hjort (1996), for example, as a reference.

On the other hand there are classification problems where the groups are themselves unknown *a priori* and the primary purpose of the data analysis is to determine the groupings from the data themselves, so that the entities within the same group are in some sense more similar than those that belong to different groups. This type of classification problem is referred to as unsupervised learning, and in statistical terminology falls under the heading of cluster analysis.

If one were to consider classification problems in three stages, input, algorithms and output, it would be fair to say that the vast majority of the work has focused on the second of these. However, it is clear that a careful thought about what variables to use and how to characterize and summarize them as inputs to methods of classification are very important issues that would involve both statistical and subject matter consideration in applications. And similarly, the most challenging aspect of most analyses of data tends not to be the choice of a particular method but the interpretation of the output and results of algorithms.

The discriminant analysis situation has been a more integral part of the historical develop-

ment of multivariate statistics, although the cluster analysis case received most of its impetus from fields such as psychology and biology until relatively recently, being nowadays used in many fields including machine learning, data mining, pattern recognition, image analysis and bioinformatics. In part, the historical lack of statistical emphasis in cluster analysis may be due to the greater inherent difficult of the technical problems associated with it. Even a precise and generally agreed upon definition of a cluster is hard to come by. The data-dependent nature of the clusters, the number of them and their composition appear to cause fundamental difficulties for formal statistical inference and distribution theory. Except for *ad hoc* algorithms for carrying out cluster analysis themselves, counterparts of many other statistical methods that exist for the discriminant analysis cases are unavailable for the cluster analysis situation. In this work we have focused on this second category of classification.

Cluster analysis is a common technique for statistical data analysis. It searches through data for observations that are similar enough to each other to be usefully identified as part of a common cluster, what is a very intuitive and natural objective. More precisely, it intends to combine observations into groups or clusters such that:

1. Each cluster is homogeneous or compact with respect to certain characteristics; that is, *observations in each group are similar to each other.*

2. Each group should be different from other groups with respect to the same characteristics; that is, *observations of one group should be different from the observations of other groups.*

Rudimentary, exploratory procedures are often quite helpful in understanding the complex nature of multivariate relationships, and searching the data for a structure of natural groupings or clusters is an important example of such techniques. If each observation is identified with one and only one cluster, then the clusters constitute a partition of the data that can be very useful for statistical purposes. For instance, it is often possible to summarize a large multivariate data

set in terms of a "typical" member of each cluster and this would be more meaningful that only looking at a single "typical" member of the entire data and much more concise than individual descriptions of each observation. Another use occurs when one is attempting to model data in the presence of cluster structure. Better results may be achieved by taking this structure into account before attempting to estimate any of the relationships that may be present. In general, cluster analysis can provide an informal means for assessing dimensionality, identifying outliers or suggesting interesting hypothesis concerning relationships. Besides the term cluster analysis there are a number of terms with similar meanings, including data clustering (or just clustering), automatic classification, numerical taxonomy, botryology and typological analysis.

Geometrically, the concept of cluster analysis is very simple. Each data can be represented as a point in a d-dimensional space, where d is the number of variables or characteristics used to describe the subjects. Cluster analysis groups observations such that each group observations are similar with respect to the clustering variables. However, graphical procedures for identifying clusters may not be feasible when we have many observations or when we have more than three variables or characteristics. What is needed, in such a case, is an analytical technique for identifying groups or clusters of points in a given dimensional space. If grouping is done on the basis of similarities or distances (dissimilarities) then the inputs required are similarity measures or data from which similarities can be computed. Cluster analysis is a heuristic technique; therefore a clustering solution or grouping will result even when there may not be any natural groups or clusters in the data, and establishing the reliability and external validity of a cluster solution is very important.

The first step in cluster analysis is to select a measure of similarity (or distance), but the definition of similarity or homogeneity varies from analysis to analysis and depends on the objectives of the study. A simple measure is Manhattan distance, equal to the sum of absolute distances for each variable. The name comes from the fact that in a two-variable case, the variables can be plotted on a grid that can be compared to city streets, and the distance

between two points is the number of blocks a person would walk. However, a more common measure is Euclidean distance, computed by finding the square of the distance between each variable, summing the squares, and finding the square root of that sum. In the two-variable case, the distance is analogous to finding the length of the hypotenuse in a triangle; that is, it is the distance "as the crow flies". A review of cluster analysis in health psychology research (Clatworthy *et al.*, 2005) found that the most common distance measure in published studies in that research area is the Euclidean distance or the squared Euclidean distance.

Once a distance measure is selected, elements in the data set can be combined and then a decision is made on the type of clustering technique to be used and on the particular clustering method. In general, data clustering algorithms can be hierarchical or non-hierarchical (also called flat or partitional). Hierarchical algorithms find successive clusters using previously established clusters, whereas flat algorithms determine all clusters at once. Hierarchical algorithms can be agglomerative, starting with each element as a separate cluster and merging them in successively larger clusters, or divisive, starting with the whole set and dividing it into successively finer clusters, to form a hierarchy that is traditionally represented as a tree data structure (called a dendrogram) with individual elements at one end and a single cluster with every element at the other. Cutting the tree at a given height will give a flat clustering at a selected precision. Some other examples of partitional clustering are k-means or fuzzy c-means.

The k-means algorithm (MacQueen, 1967) assigns each point to the cluster whose center (also called centroid) is nearest. The center is the average of all the points in the cluster – that is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster. The main advantages of this algorithm are its simplicity and speed, which allows it to run on large datasets. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments. It maximizes inter-cluster (or minimizes intra-cluster) variance, but does not ensure that the result has a global minimum variance.

In soft clustering, each point has a degree of belongingness to the clusters rather than belonging completely to just one cluster. Thus, points on the edge of a cluster, may be in the cluster to a lesser degree than points in the center of the cluster. For each point x we have a coefficient giving the degree of being in the k-th cluster $u_k(x)$. Usually, the sum of those coefficients is defined to be 1, so that $u_k(x)$ denotes a probability of belonging to a certain cluster. With the fuzzy c-means algorithm, introduced in Bezdek (1981), the centroid c_k of the k-th cluster is the mean of all points in that cluster, weighted by their degree of belongingness to the cluster. The algorithm minimizes intra-cluster variance as well, but has the same problems as k-means: the minimum is a local minimum and the results depend on the initial choice of weights.

A recent approach to clustering that has attracted a lot of attention is the spectral clustering, which usually involves taking the top eigenvectors of some (affinity) matrix S_{ij}, based on a measure of the similarity between point i and j, and using them to cluster the points. Sometimes such techniques are also used to perform dimensionality reduction for clustering in fewer dimensions. Its ability to identify non convex clusters makes it ideal for a number of applications, including bioinformatics (Pentney and Meila, 2005), software clustering (Shokoufandeh *et al.*, 2002, 2005), image segmentation (Shi and Malik, 2000) and speech recognition (Bach and Jordan, 2005; Brown and Cooke, 1994). For example, the Shi-Malik algorithm (Shi and Malik, 2000) partitions points into two sets (S_1, S_2) based on the eigenvector v corresponding to the second-smallest eigenvalue of the Laplacian $L = I - D^{1/2}SD^{1/2}$ of S, where D is the diagonal matrix $D_{ii} = \sum_j S_{ij}$. The algorithm can be used for hierarchical clustering, by repeatedly partitioning the subsets. A related algorithm is the Meila-Shi algorithm (Meila and Shi, 2001), which takes the eigenvectors corresponding to the k largest eigenvalues of the matrix $P = SD - 1$ for some k, and then invokes another algorithm (e.g. k-means) to cluster points by their respective k components in these eigenvectors. For a review of spectral clustering and the main results in the literature see for example Verma and Meila (2003).

The last step in cluster analysis is the decision regarding the number of clusters and finally, the cluster solution is interpreted. Notice that statistical theory cannot provide a complete theory of classification. We cannot say how similarities should be judged, though we can give technical assistance in constructing distances, for example. One general model is that the data is a random sample from some population with a probability distribution P. A technique produces some clusters in the sample. A theoretical model generates some clusters in the population with the distribution P. We evaluate the technique by asking how well the sample clusters agree with the population clusters.

The proliferation of clustering techniques over the past has demanded the development of ways of measuring the similarity between two clusterings, to compare how well different data clustering algorithms perform. There have been several suggestions, the most popular ones being the Rand index (Rand, 1971) and confusion matrices that summarize overlappings of clusters.

Another important question in cluster analysis is how many clusters should be sought in the data set. There is a lot of work in this direction, but the problem is still open. Some of the techniques used to determine the number of groups are Akaike's information criterion (Akaike, 1974) or the minimum description length criterion (Rissanen, 1978).

Though clustering is an old data analysis problem, microarray experiments, a revolutionary high-throughput tool for the study of gene expression, have revived the interest in cluster analysis by raising new methodological and computational challenges. The size of data bases that store genome related information increases exponentially and techniques for mining such a huge amount of data are essential in current research. The need to sort genes into groups based on some notion of similarity is present in all current genome-wide biological investigations. The hope is that similarity with respect to a measurable quantity, such as gene expression, is often indicative of similarity with respect to more fundamental qualities, such as function.

Cluster analysis techniques are extremely useful for grouping genes. Since Eisen *et al.* (1998)

published their landmark paper describing cluster analysis of gene expression data in the budding yeast Schizosaccharomyces cerevisiae, this analytical approach has become a standard for the analysis of DNA microarray data.

Though in the present work we have exclusively used real data from the field of molecular biology, this is not the only one where cluster analysis has a huge application. As interesting examples consider:

Marketing research: Cluster analysis is widely used in market research when working with multivariate data from surveys and test panels. Market researchers use cluster analysis to partition the general population of consumers into market segments and to better understand the relationships between different groups of consumers/potential customers.

Social network analysis: In the study of social networks, clustering may be used to recognize communities within large groups of people.

Image segmentation: Clustering can be used to divide a digital image into distinct regions for border detection or object recognition.

Data mining: Many data mining applications involve partitioning data items into related subsets; the marketing applications discussed above represent some examples. Another common application is the division of documents, such as World Wide Web pages, into genres.

This thesis is organized as follows. Chapter 1 is a general review of cluster analysis techniques and clustering related problems. In order to understand the nature of the real data we will be dealing with, we will describe briefly some basics of molecular biology (sections 1.2 and 1.3), and will explain how the technique of microarrays allows to obtain this data (sections 1.4 and 1.5). Some basics about graph theory and the Minimum Description Length principle, needed for the understanding of the methods presented here, are described in sections 1.6 and 1.7, respectively, while the basic features about clustering are summarized in section 1.8. The open-related problems depicted in section 1.9 include the comparison between clusters, the

refinement of iterative clustering methods that depend on the initial selection of parameters (e.g. k-means) and the choice of an appropriate number of clusters.

Each of these problems is addressed in a different chapter. In chapter 2 we propose a clustering comparison method and its implementation that finds a many-to-many correspondence between groups of clusters from two clusterings, either flat versus flat or flat versus hierarchical. In section 2.2 we use a weighted bipartite graph to represent the clusters from two flat clusterings and visualize the correspondences between them by minimizing the number of crossings in this bi-graph. The greedy algorithm used to merge clusters and to determine a correspondence between clusters from two flat clusterings is described in section 2.2.1. In section 2.3 we analyze the problem of comparing a flat and a hierarchical clusterings. We propose an algorithm that finds the optimal cut-off points in the dendrogram as this is explored first depth. To decide whether or not we cut further ahead in the dendrogram we use two different scores. In section 2.3.1 we describe the first one, based on information theory concepts and on the Minimum Description Length principle. In section 2.3.2 we use a score based on the layout aesthetics of the bi-graph representing both clusterings.

In chapter 3 we describe two methods to refine the initial state of k-means, both using a generalization of the univariate median (so-called data depth), which is more robust than the multivariate mean. The definition of data depth that we use, described in López-Pintado and Romo (2006), has the advantage of having low computational cost, as shown in section 3.2.1. In section 3.2.2 we use a simple approach, alternating the computation of the deepest point in each cluster with k-means. In section 3.2.3 we sample the data via bootstrap to obtain different versions of the data set that are clustered to finally combine information from all of them. This second method can also be used to construct a soft (non-hard) clustering technique, as explained in section 3.4.

In chapter 4 we use ideas from the previous chapters to develop two methods that estimate the number of clusters underlying in a data set. The first one is a clustering sampling method

and it is described in section 4.2.2. The basic idea is to perform successive comparisons of pairs of flat clusterings, run on the same data set. Depending on whether we compare clusterings with the same or with a different number of initial clusters we define two approaches, the single k and the mixed k, respectively. The second one, described in section 4.3, is based on finding the most representative points of the clusters obtained by some clustering algorithm, using a notion of data depth, and selecting the number of clusters that minimizes the sum of the diameters of the subsets thus obtained.

Finally, in chapter 5 we summarize the main findings contained in this thesis and enumerate some of the possible research lines derived from this work, like the comparison of two hierarchical clusterings or two soft clusterings.

1.2. Basics of molecular biology

Since the DNA structure was discovered in 1953 by James Watson and Francis Crick, the genomic field has experienced an enormous development, and it is widely believed that the thousands of genes and their products (RNA and proteins) in a given living organism work in a complicated way.

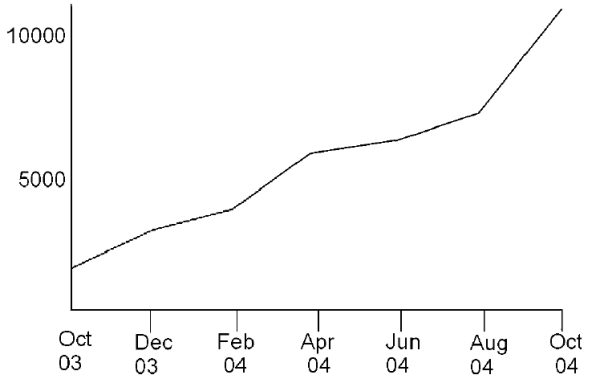


Figure 1.1: Growth of the ArrayExpress database from October 2003 to September 2004.

In the past several years, a new technology called DNA microarray has attracted a great interest among researchers of different fields, such as biologists or statisticians, because it

allows to monitor the whole genome of any organism on a single chip, so that we can have a good picture of the interactions among thousands of genes simultaneously. The potential of such technology for functional genomics is tremendous, as measuring gene expression levels in different developmental stages, different body tissues, different clinical conditions and different organisms is instrumental in understanding gene functions, gene networks or effects of medical treatments. They have led to the creation of data bases that store genetic information, and it is quite remarkable that their size grows exponentially. As an example, figure 1.1 represents the number of hybridizations submitted to ArrayExpress (http://www.ebi.ac.uk/arrayexpress/) from October 2003 to September 2004. Thus, finding relevant facts in those enormous data bases and analyzing them is essential in the Biology of this new century.

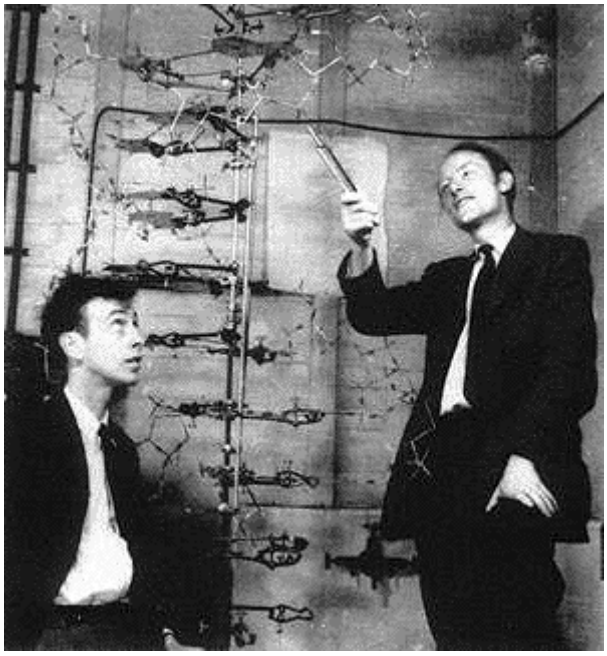


Figure 1.2: Francis Crick shows James Watson the model of DNA in their room number 103 of the Austin Wing at the Cavendish Laboratory, Cambridge, UK.

A key step in the analysis of gene expression data is the identification of groups of genes that manifest similar expression patterns over several conditions, whose corresponding algorithmic problem is to cluster multi-condition gene expression patterns. Clustering is one of the most

widely used methods in gene expression data analysis, which has revived interest in cluster analysis by raising new computational challenges.

Gene expression data is often described by a matrix with thousands of rows –representing genes– and maybe hundreds of columns –representing conditions or samples– and it is important to reduce the dimensionality of this data to understand the structure of the data and the biological information contained in it. One of the main goals of clustering methods is to discover similar profiles in the behavior of genes and represent this large amount of data by a few number of groups with similarly behaving genes.

1.3. Central dogma of molecular biology

Living organisms are complex systems. Hundreds of thousands of proteins exist inside each of them to help carry out daily functions. These proteins are produced locally, assembled to exact specifications. The enormous amount of information required to manage this complex system correctly is stored in the nucleic acids, very large molecules with two main parts: a backbone of sugar and phosphate and attached molecules called nucleotide bases. There are only four different nucleotide bases within each nucleic acid (but each one contains millions of bases). The order in which these nucleotide bases appear in the nucleic acid codes for the information carried in the molecule. Thus, a strand of nucleic acid can be described mathematically as a (very long) word constructed over an alphabet $\mathcal{A}$ of four letters.

The Desoxiribo-Nucleic Acid (DNA) stores the genetic information. Its bases are adenine (A), cytosine (C), guanine (G) and thymine (T), and therefore $\mathcal{A} = \{A, C, G, T\}$. It is a double stranded molecule formed by complementary pairs: adenine always bonds to thymine (and vice versa) and guanine always bonds to cytosine (and vice versa). The whole DNA stored in the cells is called the “genome”.

The other nucleic acid involved in the production of proteins is the Ribo-Nucleic Acid (RNA), a single-stranded molecule that contains the bases adenine, cytosine, guanine and, instead of

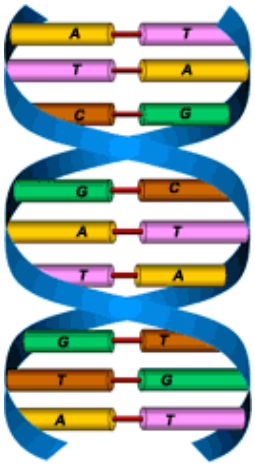


Figure 1.3: Structure of a DNA molecule.

thymine, uracil (U). Here, the alphabet is $\mathcal{A}' = \{A, C, G, U\}$. It serves as a genetic messenger, carrying the information contained in the DNA to the parts of the cell where it is used to build proteins.

Three different processes are responsible for the inheritance of genetic information:

- **Replication:** the double-stranded DNA is duplicated to give identical copies. This process perpetuates the genetic information.
- **Transcription:** a DNA segment that constitutes a gene (the basic unit of hereditary material: an ordered sequence of nucleotide bases that encodes a product, or a "meaningful" word from $\mathcal{A}$) is read and transcribed into a single stranded sequence of RNA.
- **Translation:** the RNA sequence is translated into a sequence of amino acids, the building blocks of proteins, as these are formed. During translation, three bases are read at a time from the RNA and translated into one amino acid. (The amino acids are the letters of the alphabet used to describe proteins.)

The central dogma of molecular biology, first enunciated by Crick in 1958, states that the

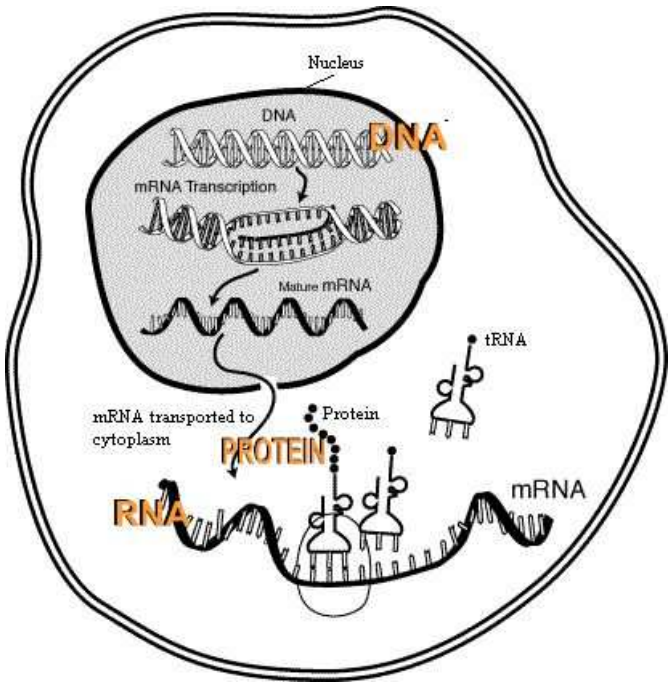


Figure 1.4: The central dogma of the molecular biology "deals with the ***detailed*** residue-by-residue transfer of sequential information. It states that such information cannot be transferred from protein to either protein or nucleic acid". In other words, once information gets into protein it cannot flow back to nucleic acid.

flow of genetic information travels from DNA to RNA, and finally to the translation of proteins, but not the other way around.

Unfortunatelly, knowing the whole sequence of a particular genome is not enough to understand how genes work (which ones are their functions), how proteins are built up, how cells conform organisms or what fails when certain malady develops. Therefore, the funtional genomics tries not only to identify the genome sequence (the whole ensemble of words from $\mathcal{A}$ that encode the genetic information) but to learn the genes functions and relationships (understand the "meaning" of these words).

1.4. Microarrays

Microarrays were first used to study global gene expresion in 1997 (DeRisi *et al.*, 1997). A microarray is a slide onto which DNA molecules are attached at fixed locations called spots. There may be tens of thousands of spots on an array, each containing tens of millions of identical DNA molecules. For gene expression studies, each of these molecules should identify a single RNA molecule in a genome.

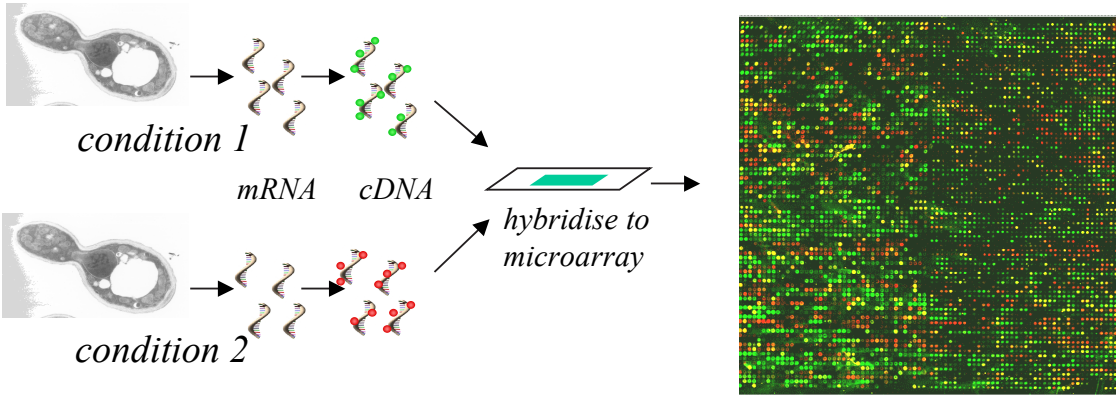


Figure 1.5: Construction of a microarray.

The most popular application of microarrays is to compare the gene expression levels in two different samples (e.g. the same cell type under two different conditions). This is based on labelling the RNA extracted from each of the samples in two different ways (for example, a green label for the sample from condition 1 and a red label for condition 2). Both extracts are incubated with the microarray; labelled gene products bind to their complementary sequence while non-bound samples are removed by washing. The hybridized microarray is excited by a laser and scanned to detect the red and green dyes. The amount of fluorescence emitted upon laser excitation corresponds to the amount of RNA bound to each spot. If the RNA from condition 1 is in abundance, the spot will be green; if the RNA from condition 2 is in abundance the spot will be red. If both are equal the spot will be yellow and if neither are present it will appear black. Thus, from the fluorescence intensities and colours for each spot, the relative expresion levels of the genes in both samples can be estimated. In this way, thousands of data

points, each providing information about the expression of a particular DNA transcript, can be obtained from a single experiment.

1.5. Data generation, processing and analysis

The raw data that is produced from microarray experiments are digital images that are analysed to obtain information about the gene expression levels. Each spot has to be identified and its intensity measured and compared to values representing the background, but this is not a trivial task and it is important to know the principles of image analysis for a better understanding of the limitations of microarray data.

The first step to process the data is to construct a spot quantitation matrix, where each row corresponds to one spot on the array and each column represents different quantitative characteristics of the spot (the total intensity of the spot, the mean, median or mode of the pixel intensity distribution, the local background, the standard deviation of both signal and background, etc.).

The data from multiple hybridizations must be transformed and organised in a gene expression matrix, where each row represents a gene and each column represents an experimental condition. Obtaining such a matrix is not either a trivial task: for example, a single gene can be represented by several spots on the array or the same experimental condition can be monitored in multiple hybridizations carried out over replicated experiments. All the quantities relating to a gene have to be combined to obtain a single number. Moreover, measurements obtained using different arrays have to be normalized to make them directly comparable. Because normalization changes the data, one has to understand the principles of the technique used and how it changes the data. Furthermore, all normalization strategies (total intensity, mean log centring, linear regression, lowess...) are based on some underlying assumptions regarding the data and consequently, the normalization approach must be appropriate to the experimental conditions that are used. There is no standard best method for microarray data normalization

fitting all cases, and new methods are being developed (see, for example, Bolstad *et al.* (2003); Cheng and Li (2005)). After the gene expression data matrix has been generated, we can begin analysing and mining it.

Microarray technology is still developing rapidly, so it is natural that there are no established standards for microarray experiments or for processing the raw data. In the absence of such standards, the Microarray Gene Expression Data (MGED) society (http://www.mged.org/) has developed recommendations for the minimum information needed about a microarray experiment, that attempts to define the set of information sufficient to interpret the experiments and the results of the experiment unambiguously, and to enable verification of the data (Brazma *et al.*, 2001).

1.6. Graph theory basics

Configuration of nodes and connections occur in a great variety of applications. They may represent physical networks (circuits) or organic molecules, or can be used to represent less tangible interactions, as may occur in sociological relationships or databases. Formally, such configurations are modelled by combinatorial structures called graphs. Graph theory defines a *graph* $G = \{N, E\}$ as a set of nodes N (or vertices) and a set of edges E, each of which joins two nodes –called its endpoints. The nodes and edges may have additional attributes, such as colour or weight. Some graph models may require graphs with direction on their edges, with multiple connections between vertices or with connection from a vertex to itself.

We introduce some basic terminology needed in next chapters.

A *multi-edge* is a collection of two or more edges having identical endpoints. A *directed edge* is an edge, one of whose endpoints is designated as the tail and whose other endpoint is designated as the head. A *directed graph* is a graph each of whose edges is directed.

A *bipartite graph* G is a graph whose node-set N can be partitioned into two subsets U and V, such that each edge of G has one endpoint in U and one endpoint in V.

A graph is called *connected* if for every pair of the nodes in the graph there is a path through the edges from one node to the other. (In a physical representation of a graph, a path models a continuous traversal along some edges and vertices.)

It has to be noticed the difference between 'structure' and 'representation'. Structure is what characterizes a graph itself and it is independent of the way the graph is represented. The possible representations include drawings, incidence tables or formal specifications, among others. One prominent aspect of the structure is the system of smaller graphs inside a graph, which are called subgraphs. Formally, a *subgraph* of a graph G is a graph H whose vertices and edges are all in G. For a given graph G, the *subgraph induced on a vertex subset* U of N_G is the subgraph of G whose node-set is U and whose edge-set consists of all edges in G that have both endpoints in U. A subset S of N_G is called a *clique* if every pair of nodes in S is joined by at least one edge, and no proper superset of S has this property. Thus, a clique of a graph is a maximal subset of mutually adjacent nodes in G.

Many optimization problems can be formulated in terms of finding a minimum (or maximum) cut in a network or graph. A *cut* is formed by partitioning the nodes of a graph into two mutually exclusive sets. An edge between a node in one set and a node in the other is said to be in the cut. The *weight*, or the capacity, *of the cut* is the sum of the weights of the edges that have exactly one endpoint in each of the two sets. The problem of finding the minimum weight cut in a graph plays an important role in the design of communication networks. If a few of the links are cut or fail, the network may still be able to transmit messages between any pair of its nodes. If enough links fail, however, there will be at least one pair of nodes that cannot communicate with each other. Thus an important measure of the reliability of a network is the minimum number of links that must fail in order for this to happen. This number is referred to as the *edge connectivity* of the network and can be found by assigning a weight of 1 to each link and finding a minimum weight cut. More precisely, we call edge connectivity of a graph the minimum number of edges whose removal results in a disconnected graph.

For further insight into graph theory concepts, see, for example Gross and Yellen (1999).

1.7. The Minimum Description Length principle

The Minimum Description Length (MDL) principle is a relatively recent method for inductive inference *that provides a generic solution to the model selection problem.* MDL is based on the following insight: any regularity in the data can be used to compress the data, i.e., to describe it using fewer symbols than the number of bits needed to describe the data literally. The more regularities there are, the more the data can be compressed.

It has several attractive properties:

1. **Occam's razor**: MDL chooses a model that trades-off goodness-of-fit on the observed data with complexity or richness of the model. As such, MDL embodies a form of Occam's Razor, a principle that is both inductively appealing and informally applied throughout all the sciences.

2. **No overfitting**: MDL procedures automatically and inherently protect against overfitting and can be used to estimate both the parameters and the structure of a model.

3. **Bayesian interpretation**: MDL is closely related to Bayesian inference, but avoids some of the interpretation difficulties of the Bayesian approach, especially in the realistic case when it is known a priori to the modeler that none of the models under consideration is true.

4. **No need for underlying truth**: in contrast to other statistical methods, MDL procedures have a clear interpretation independent of whether or not there exists some underlying true model.

5. **Predictive interpretation**: Because data compression is formally equivalent to

a form of probabilistic prediction, MDL methods can be interpreted as searching for a model with good predictive performance on unseen data.

The fundamental idea in the MDL principle is the concept of learning as "data compression". To understand it, consider the following sequences of bits, of length 10000, where we only show the beginning and the end:

$$0001000010001000100010\ldots0001000100010001000100010001 \tag{1.7.1}$$

$$0111010011010010011\ldots1010111010111011000101100010 \tag{1.7.2}$$

$$0001100000101010000\ldots0010001000010000001000110000 \tag{1.7.3}$$

The first sequence is a 2500-fold repetition of 0001. Intuitively, the sequence looks regular and it seems that there is an simple law underlying it. It might make sense to predict that future data will behave according to this law.

The second sequence has been generated by tosses of a fair coin. It is, intuitively speaking, as random as possible and there is no regularity underlying it. Indeed, we cannot find such regularity either when we look at the data.

The third sequence contains approximately four times as many 0's as 1's. It looks less regular, more random than the first, but it seems less random than the second. There is some discernible regularity in this data, but of a statistical rather than of a deterministic kind. Again, predicting that future data will behave according to the same regularity seems sensible.

We claimed that any regularity detected in the data can be used to compress the data, i.e., to describe it in a short manner. Descriptions are always relative to some description method which maps descriptions D' in a unique manner to data sets D. A particular versatile description method is a general-purpose computer language like C. A description of D is then any computer program that prints D and then halts.

Then, for sequence 1.7.1 we can write a program like:

for i=1:2500; print("0001"); halt

which prints the first sequence, but is clearly a lot shorter. Thus, sequence 1.7.1 is highly compressible.

On the other hand, it can be shown that if one generates a sequence like the second one by tosses of a fair coin, then, with extremely high probability, the shortest program that prints 1.7.2 and then halts will look like:

print("01110100110100100110 ... 10101110101110110001011000010"); halt

This program's size is about equal to the length of the sequence. Clearly, it does nothing more than repeat the sequence.

The third sequence lies in between the first two: generalizing n =10000 to arbitrary length n, it can be shown that the first sequence can be compressed to $O(log\, n)$ bits; with overwhelming probability, the second sequence cannot be compressed at all; the third sequence can be compressed to some length αn, with $0 < \alpha < 1$.

This viewing of learning like data compression can be made precise in several ways:

- The **idealized MDL** looks for the shortest program that generates the given data. This approach is not feasible in practice for the following two reasons:

  1. **uncomputability**: it can be shown that there exists no computer program that, for every set of data D, when given D as input, returns the shortest program that prints D (Li and Vitányi, 1997).
  2. **arbitrariness/dependence on syntax**: in practice we are confronted with small data samples for which the length of the shortest program written in two different languages $\mathbb{A}$ and $\mathbb{B}$ can differ substantially (Kolmogorov, 1965; Solomonoff, 1964). Then, the model chosen by idealized MDL may depend on arbitrary details of the syntax of the programming language under consideration.

- The **practical MDL** comes in a crude version based on two-part codes and in a modern, more refined version based on the concept of universal coding (Barron *et al.*, 1998). The basic ideas underlying these approaches can be found in boxes 1.7.1 and 1.7.2.

These methods are mostly applied to model selection but can also be used for other problems of inductive inference. In contrast to most existing statistical methodologies, they can be given a clear interpretation irrespective of whether or not there exists some true distribution generating data; inductive inference is seen as a search for regularity properties in interesting statistics of the data, and there is no need to assume anything outside the model and the data. In contrast to what is sometimes thought, there is no implicit belief that simpler models are more likely to be true. MDL embodies a preference for simple models, but this is best seen as a strategy for inference that can be useful even if the environment is not simple at all.

Good places to find further exploration of MDL are Barron *et al.* (1998) and Hansen and Yu (2001).

Let $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \ldots$ be a list of candidate models, each containing a set of point hypotheses. The best point hypothesis $H \in \mathcal{H}^{(1)} \cup \mathcal{H}^{(2)} \cup \ldots$ to explain the data D is the one which minimizes the sum $L(H) + L(\mathrm{D}|\, H)$, where:

- $L(H)$ is the length, in bits, of the description of the hypothesis, and
- $L(\mathrm{D}|\, H)$ is the length, in bits, of the description of the data when encoded with the help of the hypothesis.

The best model to explain D is the smallest model containing the selected H.

Box 1.7.1. Crude, two-part version of MDL principle.

Suppose we plan to select between models Let $\mathcal{M}^{(1)}, \mathcal{M}^{(2)}, \ldots$ for data $D = (x_1, \ldots, x_n)$. MDL tells us to design a universal code $\bar{P}$ for X^n, in which the index k of $\mathcal{M}^{(k)}$ is encoded explicitly. The resulting code has two parts, the two sub-codes being defined such that:

- All models $\mathcal{M}^{(k)}$ are treated on the same footing, as far as possible: we assign a uniform prior to these models, or, if that is not possible, a prior "close to" uniform.
- All distributions within each $\mathcal{M}^{(k)}$ are treated on the same footing, as far as possible: we use the minimax regret universal model $\bar{P}_{nml}(x^n \mid \mathcal{M}^{(k)})$. If this model is undefined or too hard to compute, we instead use a different universal model that achieves regret "close to" the minimax regret for each submodel of $\mathcal{M}^{(k)}$.

In the end, we encode data D explicitly encoding the models we want to select between and implicitly encoding any distributions contained in those models.

Box 1.7.2. Refined MDL principle for model selection.

1.8. Clustering gene expression data

The analysis of gene expression data is based on the hypothesis that there are biologically relevant patterns to be discovered in the data (for example, there may be genes that reflect specific cellular responses). The data mining process typically relies on analysis of the gene expression matrix using unsupervised or supervised methods.

Clustering belongs to unsupervised methods that can mine through data, extracting relevant information, without *a priori* information. Here, the concept of *a priori* information can be misunderstood, as some unsupervised algorithms still require some previous knowledge about the data, like knowing the number of true groups underlying the data; therefore, we will describe

unsupervised methods as those that use unlabelled data points, versus the labelled data used by supervised methods. For a given d-dimensional vector $g_i = (g_{i1}, \ldots, g_{id})$ a label is a $(d+1)$-th variable $g_{i,d+1}$ that reflects, for example, the group the vector belongs to. For unlabelled data, this information is not known.

A clustering problem consists of N elements and a characteristic vector for each element, $g_1, \ldots, g_N$. In gene expression data, elements are genes, and the vector of each gene contains its expression levels under some conditions (see section 1.5). A measure of pairwise similarity is then defined between such vectors. The goal is to partition the elements into subsets, which are called clusters, so that two criteria are satisfied:

- **Homogeneity**: elements in the same cluster are highly similar to each other;
- **Separation**: elements from different clusters have low similarity to each other.

Any clustering process requires to previously select some features, like the similarity measure or the algorithm used. Some of them are briefly described in subsections 1.8.1, 1.8.2 and 1.8.3.

1.8.1. Similarity measures

There are several similarity (or dissimilarity) measures between the vectors (genes) that are to be partitioned. Some algorithms use:

- the **Euclidean distance** between the d-dimensional vectors $X = (X_1, \ldots, X_d)$ and $Y = (Y_1, \ldots, Y_d)$, given by

$$D(X, Y) = \sqrt{\sum_{i=1}^{d} (X_i - Y_i)^2}$$

  This measure was used in Wen *et al.* (1998) to cluster the (time) expression profiles of 112 genes from rat spinal cord development experiments.

- the **chord distance** between vectors X and Y, defined as the length of the chord between the vectors of unit length having the same directions as the original ones (see figure 1.6). Its expression is given by

$$C(X,Y) = \sqrt{\sum_{i=1}^{d} \left(\frac{X_i}{|X|} - \frac{Y_i}{|Y|} \right)^2} \tag{1.8.1}$$

  For normalized vectors the chord and the Euclidean distance are the same; this is an important property, since some clustering methods require the use of Euclidean distance properties. Thus, if one normalizes the vectors, one can perform these analysis in the normalized space using Euclidean distance, which will give the same results as using chord distance in the original space.

- the **Pearson correlation coeficient** between vectors X and Y, that can be written as

$$R(X,Y) = \frac{\sum_{i=1}^{d} \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{d} \left(X_i - \bar{X}\right)^2 \sum_{i=1}^{d} \left(Y_i - \bar{Y}\right)^2}}$$

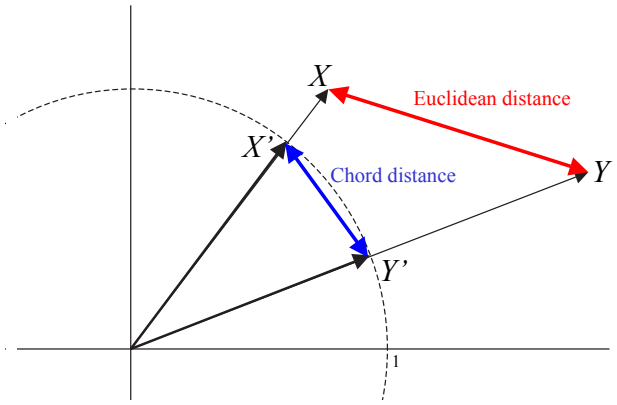


Figure 1.6: Euclidean and chord distances between bidimensional vectors X and Y.

where $\bar{X} = \frac{1}{d}\sum_{i=1}^{d} X_i$. Using the property that any dissimilarity defined by $\sqrt{(1 - s_{ij})}$ is Euclidean if the similarity matrix (s_{ij}) verifies: $0 \leq s_{ij} \leq 1$, the correlation coefficient can become an Euclidean distance with the transformation:

$$\Delta(R(X,Y),\alpha) = \sqrt{1 - \left(\frac{R(X,Y)+1}{2}\right)^{\alpha}}$$

for any $\alpha > 0$. These distances are called Linear Correlation Dissimilarities. Several authors used this coefficient; for example Eisen *et al.* (1998) used it to cluster genes from the Schizosaccharomyces cerevisiae, or Chen *et al.* (2003) studied the stress response of genes from the Schizosaccharomyces pombe.

- the **mutual information**, based on the "entropy" concept introduced in Shannon (1948), sometimes referred to as "measure of uncertainty". The entropy of a random variable is defined in terms of its probability distribution and can be shown to be a good measure of randomness or uncertainty.

  Let X be a discrete random variable taking a finite number of possible values $x_1, \ldots, x_n$ with probabilities $p_1, \ldots, p_n$ respectively, such that $p_i \geq 0, i = 1, \ldots, n$ and $\sum_{i=1}^{n} p_i = 1$. Let h be a function defined on the interval (0,1] and $h(p)$ be interpreted as the uncertainty associated with the event $\{X = x_i\}$, or as the information conveyed by revealing that X has taken on the value x_i in a given performance of the experiment. For each n, we shall define a function H_n of the n variables $p_1, \ldots, p_n$. The function $H_n(p_1, \ldots, p_n)$ is to be interpreted as the average uncertainty associated with the event $\{X = x_i\},\ i = 1, ..., n$, and it is given by

  $$H_n(p_1, \ldots, p_n) = \sum_{i=1}^{n} p_i h(p_i).$$

  Thus, $H_n(p_1, \ldots, p_n)$ is the average uncertainty removed by revealing the value of X. It can be proven (see, for example, Aczél and Daróczy (1975)) that the

expression for H_n, under some axiomatic restrictions, is

$$H_n(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i log_2 p_i. \tag{1.8.2}$$

Shannon's entropy measure of uncertainty defined by (1.8.2) satisfies many interesting properties (non-negativity, continuity, symmetry, normality, additivity, concavity, bounds on $H_n(p_1, \ldots, p_n)$, etc.). The logarithm is taken to base 2 to measure in bits.

Let, now, $X = (X_1, \ldots, X_m)$ and $Y = (Y_1, \ldots, Y_n)$ be two discrete finite random variables with joint and individual probability distributions given by $p(x_i, y_j)$, $p(x_i)$ and $p(y_j)$, respectively, and conditional probabilities denoted by $p(x_i \mid y_j)$ and $p(y_j \mid x_i)$. The joint measure of uncertainty of (X, Y) is

$$H(X, Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i, y_j) log_2 p(x_i, y_j)$$

The individual measures of uncertainty of X and Y are

$$\begin{aligned} H(X) &= -\sum_{i=1}^{m} p(x_i) log_2 p(x_i) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i, y_j) log_2 p(x_i) \\ H(Y) &= -\sum_{j=1}^{n} p(y_j) log_2 p(y_j) = -\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_i, y_j) log_2 p(y_j), \end{aligned}$$

respectively. The conditional uncertainty of Y given $X = x_i$ is

$$H(Y \mid X = x_i) = -\sum_{j=1}^{n} p(y_j \mid x_i) log_2 p(y_j \mid x_i)$$

for each $i = 1, ..., m$. The conditional uncertainty of Y given X is the average uncertainty of $H(Y \mid X = x_i)$:

$$H(Y \mid X) = \sum_{i=1}^{m} p(x_i) H(Y \mid X = x_i)$$

Similarly, we can write the conditional uncertainty of X given Y as

$$H(X \mid Y) = \sum_{j=1}^{n} p(y_j) H(X \mid Y = y_j)$$

Finally, using the expressions above, we can define the mutual information among the random variables X and Y as

$$I(X \wedge Y) = H(X) - H(X \mid Y) \tag{1.8.3}$$

that is non-negative, symmetric and equivalently expressed as $H(X) + H(Y) - H(X, Y)$. Expression (1.8.3) measures the information of X that is shared by Y. If X and Y are independent, then X contains no information about Y and vice versa, so their mutual information is zero. If X and Y are identical then all information conveyed by X is shared with Y: knowing X reveals nothing new about Y and vice versa, therefore the mutual information is the same as the information conveyed by X (or Y) alone, namely the entropy of X.

Butte and Kohane (2000) computed the entropy of gene expression patterns and the mutual information between RNA expression patterns for each pair of genes, considering that higher entropy for a gene means that its expression levels are more randomly distributed. As gene expressions are measured on a continuous scale and entropy is measured using discrete probabilities, they calculated the range of values for each gene, and then divided the range into $n = 10$ sub-ranges $x_1, \ldots, x_{10}$. Thus $p(x_i)$ represents the proportion of measurements in sub-range x_i.

On the other hand, a mutual information of zero means that the joint distribution of expression values holds no more information than genes considered separately. A higher mutual information between two genes means that one gene is non-randomly associated with the other. They hypothesized that the higher mutual information between two genes is, the more likely it is that they have a biological relationship.

However, since the degree of mutual information is dependent on how much entropy is carried by each gene expression sequence, a gene expression sequence pair exhibiting low entropies will also have low mutual information, even if they are

completely correlated. Therefore, an alternative used in Michaels *et al.* (1998) is to normalize $I(X \wedge Y)$ to the maximal entropy of each of the contributing genes, giving a high value for highly correlated sequences, independently of the individual entropies

$$I_{norm}(X \wedge Y) = \frac{I(X \wedge Y)}{max\{H(X), H(Y)\}} \tag{1.8.4}$$

Notice that the resulting measure is symmetric in its arguments but that there are other non-symmetric normalization possibilities.

Unlike Euclidean distance, this method also recognizes negatively and non-linearly correlated data sets as proximal.

1.8.2. Optimal clustering criteria

Clustering processes may result into different partitionings of the same data set depending on the specific criterion used for clustering. Often, this criterion is defined via a cost function.

Among clustering formulations that are based on minimizing a formal objective function, perhaps, the most widely used and studied is the *k-means clustering*. Given a set of N points in real d-dimensional space, and an integer k, the problem is to determine a set $\mathbf{a}$ of k points $a_1, \ldots, a_k$ in $\mathbb{R}^d$, called *centers*, so as to minimize the mean squared (Euclidean) distance from each data point to its nearest center:

$$min_{\mathbf{a} \subset \mathcal{R}^d} \sum_{i=1}^{N} min_{1 \leq j \leq k}\ d^2(x_i, a_j) \tag{1.8.5}$$

This measure is often called the squared-error distortion, or simply, *distortion*, and this type of clustering falls into the general category of variance-based clustering.

Clustering based on k-means is closely related to a number of other clustering problems. These include the *Euclidean k-medians* (or the multisource Weber problem (Weber, 1929)), in which the objective is to minimize the sum of distances to the nearest center, and the *geometric*

k-center problem, in which the objective is to minimize the maximum distance to every point to its closest center.

There are no efficient solutions known to any of these problems and some formulations are NP-hard (Garey and Johnson, 1979). Given the apparent difficulty of solving the k-means and other clustering problems exactly, it is natural to consider approximations such as heuristics, even though they make no guarantees on the quality of the results. One of the most popular heuristics for solving the k-means problem is based on a simple iterative scheme for finding a locally minimal solution. The algorithm is often called the k-means algorithm (see section 1.8.3), though there are a number of variants to this. The main algorithm is based on the simple observation that the optimal placement of a center is at the centroid of the associated cluster (Du *et al.*, 1999). However, it does not specify the initial placement of centers.

It has to be stressed that the type of clusters expected to occur in the data set (tight clusters, spread clusters, etc.) has to be taken into account to properly select the clustering criteria. Thus, we may define a good clustering criterion that will lead to a partitioning that fits well the data set.

1.8.3. Clustering algorithms

As previously outlined, there are two major approaches to clustering: hierarchical and flat (non-hierarchical). In hierarchical clustering the data is not partitioned into a particular cluster in a single step; instead, a series of partitions takes place. Hierarchical clustering is subdivided into agglomerative methods (bottom-up), which start by considering each object as a separate cluster and group the most similar objects iteratively until all the elements are included, and divisive methods (top-down), which separate successively all the observations into finer groupings. Any hierarchical clustering can be represented by a tree known as dendrogram, which illustrates the fusions or the divisions made at different stages of the analysis. In flat clustering, however, we look for a single partition of the data, where each individual belongs to a

single cluster and the set of all clusters contains all the observations in the data set. In some particular cases the "fuzzy" or "soft" clustering methods, that generate non-hard, overlapping clusters, may provide an acceptable solution. In general, the number of clusters in a flat clustering can either be provided by the user or determined by heuristic or theoretical principles. Reference monographies on clustering are Everitt (1980); Kaufman and Rousseeuw (1990) and Gordon (1999).

Widely-used hierarchical methods are:

- the **single-linkage clustering**, an agglomerative method that uses the minimum distance between objects in the two clusters as the measurement of the distance between the clusters; this leads to clusters that are spread out. The method is not sensitive to outliers.
- the **complete-linkage clustering**, also agglomerative, groups objects according to the greatest distance between the objects in the clusters. It tends to form tight clusters of similar objects, but it is sensitive to outliers.
- the **average-linkage clustering** calculates the distance between clusters as the average distance between every point in a cluster and every point in the other cluster. The method can be either weighted or unweighted, depending on whether we compensate for the size of the cluster or treat clusters of different size equally.

Many of these clustering algorithms have been applied to gene expression data. See, for example Alon *et al.* (1999) and Eisen *et al.* (1998).

The main disadvantage of hierarchical clustering is that the process of grouping continues until all the clusters are joined and, therefore, in the end, objects that have no similarity to each other are grouped together. This means that, in practice, the most relevant groupings are those that relate small number of genes. Another problem is that after each iteration of the

algorithm, there is no opportunity to re-evaluate the groupings that were assigned before. This makes hierarchical clustering less robust.

Some of the classic and novel flat clustering algorithms include:

- **k-means** (Hartigan, 1975). This is the most common method of non-hierarchical clustering. It starts with a given number of cluster centres, chosen randomly or by applying some heuristics. Next, the distance from these centroids to every object is calculated and each object is assigned to the cluster defined by the closest centroid. Then, for each cluster the new centroid is found. The distance from each object to each of the new centroids is calculated and in this way, the boundaries of the partitioning are revised. This is repeated until either the centroids stabilize –which is not guaranteed– or until an a priori defined maximum number of iterations is reached.

  The method has been applied to gene expression data, for example, in Tavazoie *et al.* (1999) and Vilo *et al.* (2000).

- **Kohonen's self-organizing maps** (Kohonen, 1990). Prior to initiating the analysis, the user defines a geometric configuration for the partitions –typically a two-dimensional rectangular or hexagonal grid– and the number of clusters. Each cluster is represented by a node called a "reference vector" and the reference vectors are placed on the chosen grid. Next, the nodes are projected onto the gene expression space and each of the data points is assigned to the nearest node, in a process known as initialization. After the initialization the following two steps are iterated:

  1. a gene is picked at random,
  2. the reference vector that is closest to the selected gene is moved closer to the randomly picked gene.

  The reference vectors that are nearby on the two dimensional grid are also adjusted

by small amounts so that they are also more similar to the randomly selected gene. Finally, the genes are mapped to the relevant partitions depending on the reference vector to which they are most similar.

As with k-means, the user has to rely on some other source of information, possibly empirical, to determine the number of clusters that best represent the available data. SOM's were first used to analyze gene expression data in Tamayo *et al.* (1999) and in Törönen *et al.* (1999).

- the **partitioning around medoids** algorithm (PAM) (Kaufman and Rousseeuw, 1990). It is based on finding k representative objects, called medoids, among the observations to be clustered so that the sum of the dissimilarities of the observations to their closest medoid is minimized. The algorithm first searches a good initial set of medoids and then finds a local minimum for the objective function, that is, a solution such that there is no single switch of an observation with a medoid that will decrease the objective. The method tends to be more robust and computationally efficient than k-means. In addition, it provides a graphical display, the silhouette plot, that can be used to select the number of clusters and to assess how well individual observations are clustered.

  The method has been used in Dudoit and Fridlyand (2002) to cluster data from human leukemia samples.

  The algorithm has been generalized to produce the Clustering LARge Applications (CLARA) algorithm and it can deal with much larger datasets. This is achieved by considering sub-datasets of fixed size. Then each sub-dataset is partitioned into k clusters using the same algorithm as in PAM. Once k representative objects have been selected from the sub-dataset, each observation of the entire dataset is assigned to the nearest medoid.

- the **fuzzy c-means** considers degrees of belongingness of points to clusters. This degree is related to the inverse of the distance of point x to the cluster:

$$u_k(x) = \frac{1}{d(center_k, x)}.$$

The coefficients are normalized and fuzzyfied with a real parameter $m > 1$ so that their sum is 1. So

$$u_k(x) = \frac{1}{\sum_j \left(\frac{d(center_k, x)}{d(center_j, x)} \right)^{\frac{1}{m-1}}}$$

For m equal to 2, this is equivalent to normalising the coefficient linearly to make their sum 1. When m is close to 1, the cluster center closest to the point is given much more weight than the others.

The fuzzy c-means algorithm is very similar to the k-means algorithm. It begins by choosing a number of clusters; next, coefficients for being in the clusters are randomly assigned to each point. This is repeated until convergence. The centroid for each cluster is computed as $c_k = \frac{\sum_x u_k(x)x}{\sum_x u_k(x)}$, and for each point, its coefficients of being in the clusters are recomputed.

Clustering is a well-established, still growing field, and many algorithms have been designed specifically for gene expression profile clustering, like:

- the **Quality Threshold Clustering** (Heyer *et al.*, 1999), that requires more computing power than k-means, but does not require specifying the number of clusters a priori, and always returns the same result when run several times.

The QT algorithm consists on choosing a maximum diameter for clusters, and building a candidate cluster for each point by including the closest point, the next closest, and so on, until the diameter of the cluster surpasses the threshold. Next,

the candidate cluster with the most points as the first true cluster is saved, all points in the cluster are removed from further consideration, and the procedure is repeated with the reduced set of points.

- the **gene shaving method** (Tibshirani *et al.*, 1999), that relies on an iterative heuristics algorithm that identifies subsets of genes with coherent expression patterns and large variation across conditions. The algorithm works alternating Principal Component Analysis and picking the best clusters consisting of genes contributing to the most variability of data in each cycle. The algorithm starts by finding the first principal component in the gene space. Then, starting from genes that are most similar to the first principal component, the algorithm builds a mutually inclusive system of clusters of various sizes including these genes. For each of the inclusive clusters it estimates the "quality" of the cluster using a measure called "gap statistics" and chooses the cluster with the highest score. Next, the algorithm transforms the gene space by removing the component from each gene that is parallel to the first principal axis.

  Notice that gene shaving does not produce a partitioning, as a gene may belong to several clusters.

- the **CLICK algorithm** (Sharan and Shamir, 2000), that is based on a representation of the gene expression matrix as a graph and does not require any prior assumptions about the structure or the number of clusters. The initial graph is defined by edges connecting two genes in the graph if the distance between these genes is below a particular threshold. Each edge is weighted according to the similarity between the genes. The underlying hypothesis is that the true clusters correspond to approximate cliques in the graph (see section 1.6). The CLICK algorithm looks for the approximate cliques, that is, for subgraphs that can be transformed into cliques by adding a small number of edges. In each step the algorithm handles some

connected component of the subgraph induced by the yet-unclustered elements. If the component contains a single vertex, then this vertex is considered a singleton and is handled separately. Otherwise, a stopping criterion is checked. If the component satisfies this criterion, it is declared a kernel. Otherwise, the component is split according to a minimum weight cut. The algorithm outputs a list of kernels which serve as a basis for the eventual clusters (each kernel is expanded into a cluster by adding the closest singleton objects).

- **Self-Organizing Tree Algorithm** (SOTA) (Dopazo and Carazo, 1997; Herrero *et al.*, 2001), which is one of the newest methods; it combines ideas taken from hierarchical clustering and self-organizing maps to produce a clustering more robust than classical hierarchical clustering, as well as more efficient. The SOTA produces a hierarchical, binary organization of the data and proceeds divisively. A criterion, based on the comparison of the data with a randomized sample of the same data is provided as statistical support for the clustering obtained and to assist in calculating the point at which the data should no longer be subdivided into individual clusters.

It has to be noticed that using different methods to cluster a data set, or that using different parameters within the same method (for instance, the distance measure, the number k of flat clusters obtained or the way of joining clusters in agglomerative clusterings) will commonly lead to different results. Furthermore, the use of certain methods will still produce clusters, even if they are not meaningful (Quackenbush, 2001).

1.9. Open clustering-related problems

1.9.1. Comparison of two different partitions

Despite the enormous effort done on the field, there is no compelling evidence that more sophisticated clustering algorithms perform better than the simplest ones with respect to the biological insights that have been obtained, and at the same time, this does not mean that all clustering algorithms are appropriate for all data sets (Causton *et al.*, 2003). Therefore we are likely to be dealing with different clustering results from the same data sets and it would be interesting to find a way to compare and make sense out of them.

In a real case, we can be facing any of the following three cases: the comparison between

a) two flat clusterings

b) a flat and a hierarchical clustering

c) two hierarchical clusterings

The problem of comparing two different partitions of a finite set of objects has attracted substantial interest in the clustering literature (see, for example, Arabie and Boorman (1973); Fowlkes and Mallows (1983); Hubert and Arabie (1985); Hart *et al.* (2005)). The most popular alternative for comparing partitions is the Rand index RI (Rand, 1971), which is constructed as follows: if there are N observations in the data set $\mathbf{X}$ and these are divided into two partitions $\{\mathrm{A}_1, \ldots \mathrm{A}_\mathrm{m}\}$ and $\{\mathrm{B}_1, \ldots \mathrm{B}_\mathrm{n}\}$, then let N_{ij} denote the number of objects that are both in clusters A_i and B_j, and $N_{i\cdot}$ and $N_{\cdot j}$ denote the sizes of clusters A_i and B_j, respectively. Then,

$$RI = 1 + \frac{2\sum_{i,j}\binom{N_{ij}}{2} - \sum_{i}\binom{N_{i\cdot}}{2} - \sum_{j}\binom{N_{\cdot j}}{2}}{\binom{N}{2}} \tag{1.9.6}$$

However, this index is not "corrected for chance" in the sense that the index would take on some constant value (e.g. zero) under an appropriate null model of how the partitions

$\{\mathrm{A}_1, \ldots \mathrm{A}_\mathrm{m}\}$ and $\{\mathrm{B}_1, \ldots \mathrm{B}_\mathrm{n}\}$ have been chosen. Consequently, the relative sizes for this index are difficult to evaluate and compare, since it is not either a measure of departure from a common baseline nor it is normalized to lie within certain fixed bounds. To eliminate these scaling difficulties, an adjusted Rand index (ARI), computed using the general form

$$\frac{\text{index} - \text{expected index}}{\text{maximum index} - \text{expected index}}$$

was given to correct the Rand index for chance (Hubert and Arabie, 1985)

$$ARI = \frac{\sum_{i,j}\binom{N_{ij}}{2} - \sum_i \binom{N_{i\cdot}}{2}\sum_j \binom{N_{\cdot j}}{2}/\binom{N}{2}}{\frac{1}{2}\left[\sum_i \binom{N_{i\cdot}}{2} + \sum_j \binom{N_{\cdot j}}{2}\right] - \sum_i \binom{N_{i\cdot}}{2}\sum_j \binom{N_{\cdot j}}{2}/\binom{N}{2}} \tag{1.9.7}$$

which is bounded above 1 and takes on the value 0 when the index equals its expected value.

An alternative is to use confusion matrices to measure the nature and the degree of similarity between partitions. A confusion matrix effectively summarizes pairwise intersections between clusters derived from two clustering results. The similarities are then quantified by applying scoring functions to this purpose:

- the asymmetric **normalized mutual information**, NMI, given by

$$\mathrm{NMI}(\mathbf{A}, \mathbf{B}) = \frac{I(\mathbf{A} \wedge \mathbf{B})}{H(\mathbf{A})} = \frac{H(\mathbf{A}) + H(\mathbf{B}) - H(\mathbf{A}, \mathrm{B})}{H(\mathbf{A})}$$

  where

$$\begin{aligned} H(\mathbf{A}) &= \sum_{i \in \text{partitions}} p_i log_2 p_i \\ p_i &= \frac{1}{N}\sum_j \mid \mathrm{A_i} \cap \mathrm{B_j} \mid \\ H(\mathbf{A}, \mathbf{B}) &= \sum_{i,j} \frac{\mid \mathrm{A_i} \cap \mathrm{B_j} \mid}{N} log_2 \frac{\mid \mathrm{A_i} \cap \mathrm{B_j} \mid}{N} \end{aligned}$$

  which measures the amount of information shared between the two clustering results $\mathbf{A} = \{\mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{m}\}$ and $\mathbf{B} = \{\mathrm{B}_1, \ldots, \mathrm{B}_\mathrm{n}\}$ (Forbes, 1995),

- a **linear assignment** method (Gusfield, 2002), which quantifies the similarity of two clusterings by finding the optimal pairing of clusters between the two clusterings results, and measuring the degree of agreement across this pairing,

- the **receiver operator characteristic** analysis (ROC) (Swets, 1988), that enables one to quantify the distinctness of a given cluster relative to another cluster or relative to all non-cluster members.

All these techniques provide methods to quantitatively assess the overall similarities between two different clustering results, but does not provide an effective correspondence between them. Later in this work, we will describe a method that does construct a mapping between the clusters of two different clustering results, either flat vs flat or flat vs hierarchical (see chapter 2).

1.9.2. Refinement of the initial state of iterative clustering methods

As previously said, clustering refers to the task of partitioning unlabelled data into meaningful clusters. In the literature, iterative refinement is widely adopted by various unsupervised learning algorithms, such as k-means. These algorithms work as follows:

1. Initialize the parameters of the algorithm to a "current model".

2. Decide memberships of the data points to clusters, assuming that the "current model" is correct.

3. Re-estimate the parameters of the "current model" assuming that the data memberships obtained in 2. are correct, producing a "new model".

4. If the "current model" and the "new model" are close enough, stop, otherwise, go to 2.

Given an initial condition on step 1., the algorithm defines a deterministic solution. k-means and other iterative methods converge to a point that is locally maximal for the likelihood of the data given the model. Thus, these methods are sensitive to the initial point choice, and different runs of the same algorithm on the same dataset often produce different results. Therefore, the study of initialization methods is of great value for the clustering research. Some previous work on this issue can be found, for example, in Bradley and Fayyad (1998) and He *et al.* (2004).

1.9.3. Determination of the number of true groups

Another problem that has an enormous importance in many clustering methods is the determination of the number of true clusters underlying in the data, if any. Because applying a clustering algorithm to a particular data set results in a partitioning of the data, whether or not there is an underlying structure of groups, accurately estimating this number is one of the essential, non-trivial aspects of the clustering analysis. There is previous work on this matter: a summary of classic methods is given in Gordon (1999). A first approach used to face this problem was to plot an error measure for a clustering procedure versus the number of clusters used, and select the value at which the error measure stops decreasing. A study of such methods is described in Sugar (1998) and in Milligan and Cooper (1985). Indices taking into account the between and within cluster sum of squares were given, for example, by Calinski and Harabasz (1974) and Krzanowski and Lai (1985). Other methods are the silhouette statistic, proposed by Kaufman and Rousseeuw (1990), the gap statistic, by Tibshirani *et al.* (2001), the CLEST method, by Dudoit and Fridlyand (2002), the jump method, based on information theory ideas, by Sugar and James (2003), the Relative Depth statstic (ReD), based on data depth, by Jörnsten (2004), or methods based on spectral clustering, like the one proposed by Sanguinetti *et al.* (2005). There are, however, methods that do not require to specify the number of clusters since this number is found by the method itself. See for example Peña *et al.*

(2004) and Sharan and Shamir (2000).

The following three chapters in this dissertation are dedicated to these problems. The main contributions are:

- an algorithm to analyze and visualize many-to-many relationships between clusters from two different clusterings, either two flat clusterings or a flat and a hierarchical clustering;
- an analytical manner to decide where to cut a dendrogram when compared to a flat clustering. This procedure allows to cut different branches at different heights;
- a technique to refine k-means, based on alternating the search of deepest points in a cluster with the assignment of the observations to the closest deepest point;
- an alternative refinement of k-means, based on combining ideas from data depth and bootstrap replications;
- a soft-clustering method, based on generalized Voronoi regions, to provide a degree of belongingness of points to soft clusters;
- two procedures to estimate the number of groups underlying in a data set, based on the comparison of clusterings and on the search of deepest points within clusters, respectively.

Chapter 2

Comparison of clustering results

Summary

Clustering is one of the most widely used methods in gene expression data analysis, but its results are often rather sensitive to the particular clustering method and the parameters used (Quackenbush, 2001). It is important to be able to see how clusters in one clustering relate to those in another, in order to make sense out of them. In this chapter, instead of examining simple one-to-one or one-to-many relationships between the clusters in each clustering, we present a clustering comparison method and its implementation that finds a many-to-many correspondence between groups of clusters from two clusterings. The number of clusters in each of the clusterings may be different, and we allow the comparison of either two flat clusterings, or a flat and a hierarchical clustering. In the latter case, we use a mutual information-based criterion and the graph layout aesthetics to find the optimal cut-off level in different branches of the hierarchical clustering tree.

We test our method on simulated and real gene expression data. The algorithm, the user interface, and the visualization method are implemented as a part of Expression Profiler, the online microarray data analysis tool at the EBI (http://www.ebi.ac.uk/expressionprofiler).

2.1. Introduction

Clustering is one of the most widely used methods in gene expression data analysis. Clustering results are often rather sensitive to the particular clustering method and the parameters used (Quackenbush, 2001). Many 'classic' and new clustering algorithms, including ones for hierarchical and flat (e.g., k-means) clustering have been implemented in different tools and are widely used. In the absence of clear biologically meaningful benchmark clusters, it is difficult to judge which clustering method is most appropriate in a particular situation. Various cluster quality assessment methods have been proposed (e.g. Rand (1971); Fowlkes and Mallows (1983); Kaufman and Rousseeuw (1990)), but none of them can be applied universally to all datasets. Moreover, clustering is only one step in gene expression data analysis – to understand the biological meaning of different clusterings it is important to be able to compare the results obtained by different methods and to visualize the results of such comparisons. It is important to be able to see how clusters in one clustering relate to those in another, in order to make sound decisions regarding their biological meaning.

If the results of two clusterings are very different, examining simple one-to-one or one-to-many relationships between the clusters in each clustering may not always help to understand the global relationships. In this case finding the best many-to-many relationships may be more appropriate (for instance, two clusters on one side may correspond to three clusters on the other side). If we assume that there is a 'true' but unknown grouping of genes underlying the data, such many-to-many relationships between clusters may help us to approximate the 'true' clusters, via the k-means clustering algorithm with k larger than the true number of different groups.

An even more complicated situation occurs when comparing a flat to a hierarchical clustering – on the one side, we have a fixed number of clusters, but on the other side a dendrogram (e.g., see Figure 2.1). Of course, one could cut the dendrogram at some level to obtain a set of flat

clusters (Eisen *et al.*, 1998; Tibshirani *et al.*, 1999). However, how can we determine the optimal cut-off level? Furthermore, the optimal cut-off level may not be the same in all branches: in fact, as will be shown later, this occurs only in special cases. Ideally, the clustering comparison algorithm itself should determine the optimal cut-off level for each branch.

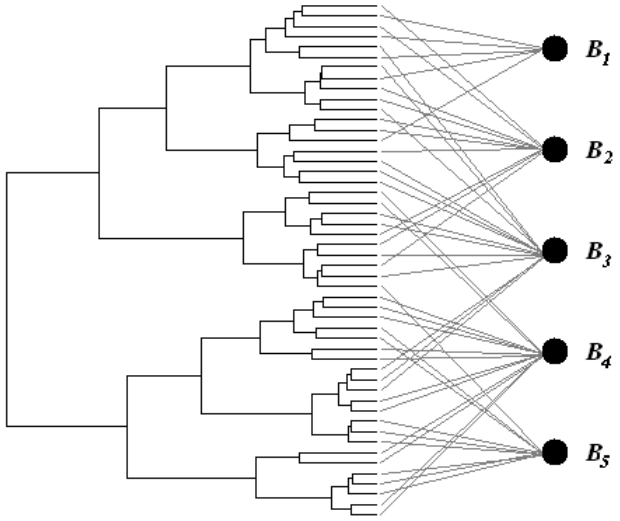


Figure 2.1: Comparison between a hierarchical clustering, represented by a dendrogram, on the left, and a flat clustering, represented by a set of nodes $B_1, \ldots, B_5$, on the right. The gray lines show in which flat cluster each leaf on the dendrogram is located.

How do we define "optimal" correspondence between clusters? One way is based on the concept of mutual information – how much one clustering can tell about the other. In terms of the minimal description length (MDL) principle (Rissanen, 1978), this can be defined as the length of the shortest encoding of one clustering using the other as given. An alternative definition is based on applying aesthetics to the visualization of the correspondence between the clusterings. We can use a bipartite graph (bi-graph) to visualize the two clusterings (each clustering is represented by a set of nodes on either side with the edges weighted by the number of common elements in the connected clusters, as in Figure 2.2), minimizing the number of edge crossings.

In next sections, we present a clustering comparison method and its implementation that

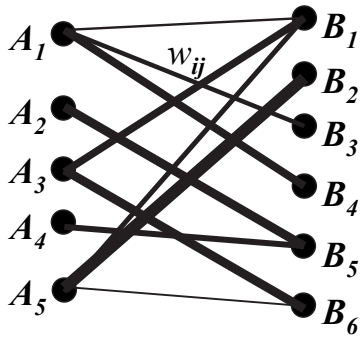


Figure 2.2: Representation of the relationships between two clustering results by means of a weighted graph. Clustering **A** has five clusters $A_1, ..., A_5$ and clustering **B** has six clusters $B_1, ..., B_6$; w_{ij} represents the weight of edge connecting nodes A_i and B_j.

finds a many-to-many correspondence between groups of clusters from two clusterings. The number of clusters in each of the clusterings may be different, and we allow the comparison of either two flat clusterings (section 2.2), or a flat and a hierarchical clustering (section 2.3). In the latter case, the mutual information-based criterion and the graph layout aesthetics are used to find the optimal cut-off level in different branches of the hierarchical clustering tree.

We have extensively tested our method on simulated and real gene expression data. In section 2.4.1 we demonstrate that, from simulated data, our method can be used to approximate true clusters without *a priori* knowledge of the number of the clusters, even with a high level of noise. In section 2.4.2 we used real gene expression data from *Schizosaccharomyces pombe* stress response microarray experiments (Chen *et al.*, 2003), and show that we can restore biologically meaningful clusters.

In section 2.5 we present the algorithm, the user interface, and the visualization method that have been implemented as a part of Expression Profiler (Kapushesky *et al.*, 2004), the online microarray data analysis tool at the EBI (http://www.ebi.ac.uk/expressionprofiler). The software, as well as the R code (see Appendix), are available from sourceforge (http://ep-sf.sourceforge.net).

2.2. Comparison of two flat clusterings

Let $\mathbf{A} = \{A_1, \ldots, A_m\}$ and $\mathbf{B} = \{B_1, \ldots, B_n\}$ be two partitions of a set of the same elements $G = \{g_1, \ldots, g_N\}$, i.e., $A_1 \cup \ldots \cup A_m = B_1 \cup \ldots \cup B_n = G$. For instance, A and B can be two different clusterings of a set of genes.

We can visualize the similarities between the clusters by drawing a weighted bi-graph, where nodes on the left and right hand sides represent, respectively, sets $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$, and the weight w_{ij} of the edge $< A_i, B_j >$ (visualized as thickness) corresponds to the number of elements common to both sets, i.e., $|A_i \cap B_j|$. In Figure 2.3.a, we have two partitions with 5 and 6 clusters each.

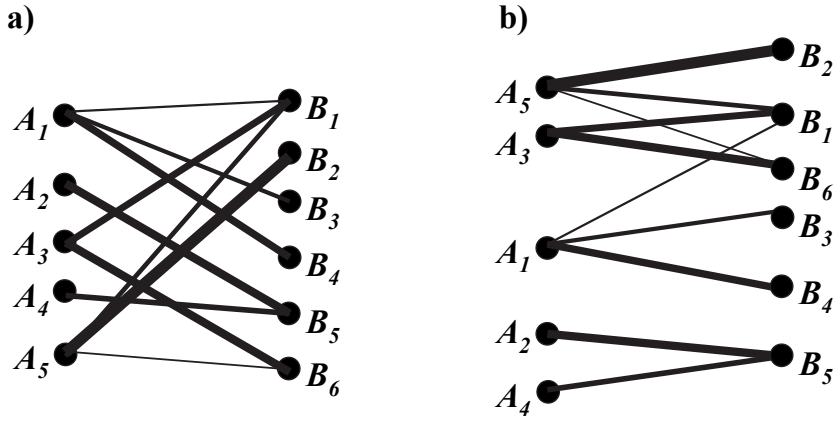


Figure 2.3: a) A weighted bi-graph representing two clusterings $A_1, \ldots, A_5$ and $B_1, \ldots, B_6$ of the same set of elements. For instance, clusters A_5 and B_2 have high overlap, while A_5 and B_5 do not have any. b) A drawing of the graph after minimizing edge crossings with the proposed gravity-centre algorithm.

If we have partitions with many clusters, or with densely interconnected ones, the bi-graph representing their correspondences can present many edge crossings, making the visualization difficult to interpret. To help see and understand the similarities between both partitions we can use a different representation of this graph, finding a layout that minimizes the weighted edge crossings.

Minimizing the number of edge crossings is a well-known problem in drawing bipartite graphs. In particular, it can be described as follows. Given a bi-graph $G = (\{\mathbf{A}, \mathbf{B}\}, \mathbf{E})$, where **A** and **B** are the sets of nodes and **E** is the set of edges linking nodes from both sets, let G be embedded in the plane, so that the nodes in **A** occupy distinct positions on the line x=0, and nodes in **B** occupy distinct positions on the line x=1, and the edges are straight lines. For a specific embedding $f(G)$, the crossing number $C_f(G)$ is the number of intersections induced by f. This depends only on the permutation of **A** and **B** along their corresponding lines and not on the specific ordinate. The minimum $C(G) = min_f C_f(G)$ was proven to be NP-hard (Garey and Johnson, 1983; Eades *et al.*, 1986).

We generalize this problem to the case of having weighted edges. and use heuristics based on those described in Gansner *et al.* (1993).

Description of the gravity-centre algorithm

The algorithm that we use assigns to each node v coordinates (x(v), y(v)). Nodes in the same partitions have the same abscissa; typically, if v∈**A**, then the abscissa equals to 0 and it equals 1, otherwise. In the initial set-up, nodes on each side are equally spaced vertically (see figure 2.3.a); the initial ordering is set by the user; by default, it will be the ordering given by the labels of the clusters in each partition.

As the iterations of the method run, the ordinates are updated. Each iteration works alternatively on either side; for a given partition, each node is assigned a new position using the barycentre (gravity-centre) function. We consider weighted edges as multi-edges collapsed into one; therefore, if node A_i is connected to B_j with an edge $< A_i, B_j >$ having weight w_{ij}, it can be understood as A_i being connected, with w_{ij} non-weighted edges, to w_{ij} nodes drawn at the same position. Then, if P_v is the list of positions (ordinates) of the incident vertices of node v, considering each position as many times as the corresponding edge weight, the gravity centre

of v is the mean over all values in P:

$$\text{gravity} - \text{centre(v)} = \frac{1}{\mid \mathrm{P_v} \mid} \sum_{\mathrm{y} \in \mathrm{P_v}} \mathrm{y}$$

Nodes on the corresponding side are reordered by sorting on these gravity centres, and drawn at points with y-coordinates equal to these values. If the new positions cause two neighbouring nodes to be less than s_{min} apart, the ordinates of the second and all the following ones are increased by s_{min}, thus shifting all nodes down.

An additional heuristic is used to improve the quality of the method: adjacent nodes are swapped if the transposition leads to a reduction of the number of edge crossings. These exchanges are repeated until there is no improvement. At each iteration, if the number of crossings decreases, the new ordering is stored. The maximum number of iterations is set to 24, as suggested in Gansner *et al.* (1993). The algorithm runs until there are no changes in the ordering of both sides or until the maximum number of iterations is reached. Figure 2.3.b shows the performance of the algorithm on the bi-graph on the left. The weighted number of edge crossings has dropped from 580 to 69 and the thickest edges do not cross each other.

Example 2.1 *Model 1*

To run the gravity-centre algorithm on simulated data we generated a number of "true" clusters as elements scattered around gravity centres with normal distribution and differently chosen standard deviations (called "noise level"). More precisely, the groups were constructed around random seeds, adding a normal random noise of mean 0 and standard deviation $\sigma = 1, \ldots, 10$, to each component, and using the following models, all in 10 dimensions:

a) four groups of size randomly chosen between 1 and 250 elements, to allow for outliers;

b) six groups of size randomly chosen between 1 and 250;

c) eight groups of size randomly chosen between 1 and 250.

We will, refer to this model as Model 1 with cluster size of 250, unless we explicitly say otherwise.

Then we ran k-means with different numbers of clusters, ran the gravity centre algorithm on the resulting clusterings 100 times and computed the average initial number of crossings and the average final number of crossings. The results for k-means with 8 and 11 clusters are shown in table 2.1.

Noise	4 groups			6 groups			8 groups		
σ	Initial	Final	Ratio	Initial	Final	Ratio	Initial	Final	Ratio
1	52308.15	2671.94	0.0511	121354.20	2213.23	0.0182	201592.30	1116.02	0.0055
2	65355.89	2724.38	0.0417	124212.90	1844.24	0.0148	205505.00	998.03	0.0049
3	56981.23	2768.42	0.0486	130970.00	2467.06	0.0188	202447.60	2246.02	0.0111
4	58789.62	4072.01	0.0693	133168.40	4550.97	0.0341	203991.86	6274.04	0.0308
5	58905.21	5320.93	0.0903	133876.80	8456.41	0.0632	236021.50	10904.52	0.0462
6	54199.81	6767.74	0.1249	121265.30	12345.30	0.1018	215593.50	20648.89	0.0958
7	56496.03	9681.38	0.1714	125582.40	19429.41	0.1547	230811.90	31793.19	0.1377
8	50083.63	11495.63	0.2295	119570.70	23513.20	0.1966	215072.30	40436.51	0.1880
9	55935.79	13747.31	0.2458	122111.30	30519.29	0.2499	213744.90	49632.38	0.2322
10	53570.23	14824.48	0.2767	118142.60	33390.28	0.2826	207067.90	59239.15	0.2861

Table 2.1: Average number of crossings before and after running 100 times the gravity-centre algorithm for pairs of two flat clusterings with 8 and 11 clusters respectively, on Model 1. The number of true groups is 4, 6 and 8.

The ratio rt between the final and the initial number of crossings can be seen as a measure of the similarity between the two clusterings that are being compared. The smaller the ratio, the more similar both clusterings are. This ratio is highly related to the percentage of elements in the data set that are closer to some other seed than the one that generated it.

Notice that this percentage serves to estimate the probability P of an element g of being closer to other seed than its own, by sampling. The estimation for 4, 6 and 8 groups and noise levels ranging between 1 and 10 is shown in Figure 2.4. Obviously, the larger the number of

seeds, the more likely an element in a noisy data set is closer to other seed than its own.

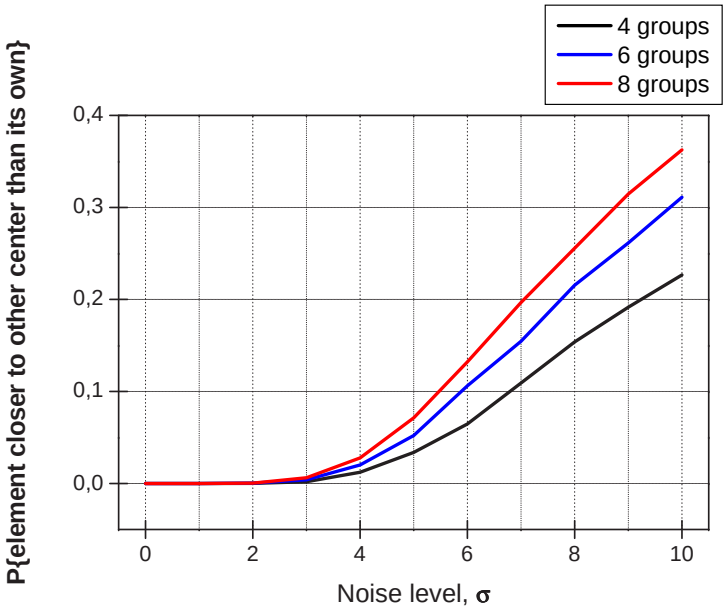


Figure 2.4: Estimation of the probability of an element being closer to some other seed than the one that generated it, on Model 1, with 4, 6 and 8 true groups.

In effect, we observed a linear relationship between this estimated probability $\hat{P}$ (the average over 100 data sets of the observed proportions of elements closer to other seeds than the ones that generated them, and the ratio rt of final and initial number of crossings in the gravity-centre algorithm. In particular, for 4 true groups, we found a correlation coefficient of 0.995201, and a linear regression given by: $P = 0{,}952073 \times rt - 0{,}049063$; for 6 true groups, the correlation coefficient was 0.999059 and the regression line: $P = 1{,}152660 \times rt - 0{,}018249$; finally, for 8 true groups, the correlation coefficient was 0.99706 and the linear model is given by $P = 1,33924 \times rt - 0{,}002331$.

Thus, for a given data set, knowing the ratio between the final and the initial number of crossings after running the gravity- centre algorithm can help to obtain some information about the proportion of elements that are likely to be misclassified, or error rate, in a clustering procedure. ■

Example 2.2 *Model 2*

We generated 3 clusters in 2 dimensions, as described in Model 2 in Dudoit and Fridlyand (2002). The clusters are independent bivariate normal random variables with $25, 25$ and 50 observations, centered at $(0, 0)$, $(0, 5)$ and $(5, -3)$. They set the covariance matrix to be the identity matrix I_2, but here we extend the model to have covariance matrix σI_2, with $\sigma \in \{0.5, 1, 1.5, \ldots, 6\}$.

We ran k-means with a different number of clusters; the results for 5 and 7 clusters are shown in table 2.2.

σ	Initial	Final	Ratio
0.5	1840.23	35.48	0.0193
1	1881.37	37.27	0.0198
1.5	1948.07	87.35	0.0448
2	1904.42	133.63	0.0702
2.5	1875.04	137.48	0.0733
3	1838.75	147.59	0.0803
3.5	1881.38	154.87	0.0823
4	1856.57	168.45	0.0907
4.5	1813.85	175.88	0.0970
5	1734.00	172.26	0.0993
5.5	1828.87	176.60	0.0966
6	1910.00	194.46	0.1018

Table 2.2: Average number of crossings before and after running 100 times the gravity-centre algorithm for pairs of two flat clusterings with 5 and 7 clusters respectively, in Model 2. The number of true groups is 3.

The estimation of the error rate is shown in Figure 2.5. Again, the correlation coefficient between both parameters is high (0.948607) and the regression line for the error rate is given by $P = 4{,}8174 \times rt - 13{,}4621$. ■

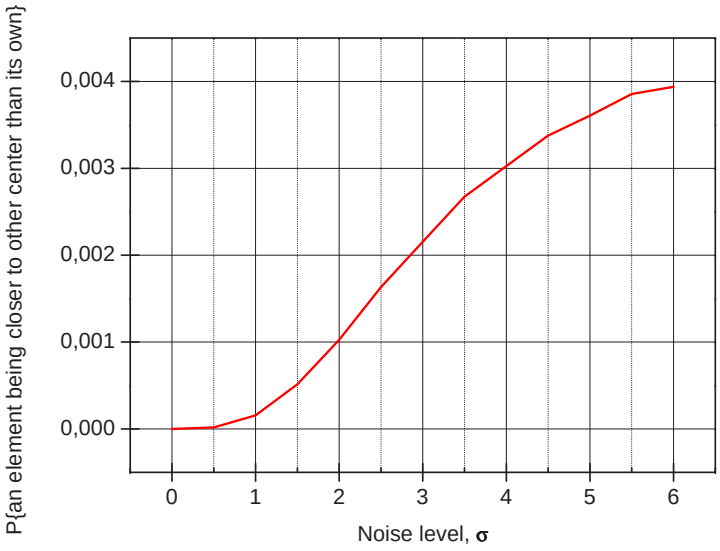


Figure 2.5: Estimation of the probability of an element being closer to some other seed than the one that generated it on Model 2.

2.2.1. Merging clusters. The greedy algorithm

This procedure, however, does not allow to construct a mapping between clusters or sets of clusters from either clustering. Instead, it provides a visualization of the relationships between clusters that can help to have a general overview about the similarities of both clusterings. To effectively find and visualize the most important many-to-many correspondences between the two partitions, we additionally use a simplified representation of the bi-graph.

The basic idea is the following: we want to group the sets A_i (more precisely, the elements g of the sets A_i) into groups $X_1, \ldots, X_k$ (i.e., for each $i = 1, \ldots, m$, every $g \in A_i$ belongs to exactly one X_j, for $j = 1, \ldots, k$) and the sets B_i into groups $Y_1, \ldots, Y_k$ (i.e., for each $i = 1, \ldots, n$, every $g \in B_i$ belongs to exactly one Y_j, for $j = 1, ..k$), so that

$$X_1 \approx Y_1, \ldots, X_k \approx Y_k$$

$X \approx Y$ means here that X and Y have high overlap, i.e., most elements on one side are also present on the other side. For instance in Figure 2.6 below, we form groups $X_1 = A_1 \cup A_2$,

$X_2 = A_3, X_3 = A_4$ and $Y_1 = B_1, Y_2 = B_2 \cup B_3$, and $Y_3 = B_4$, because $A_1 \cup A_2 \approx B_1, A_3 \approx B_2 \cup B_3$, and $A_4 \approx B_4$.

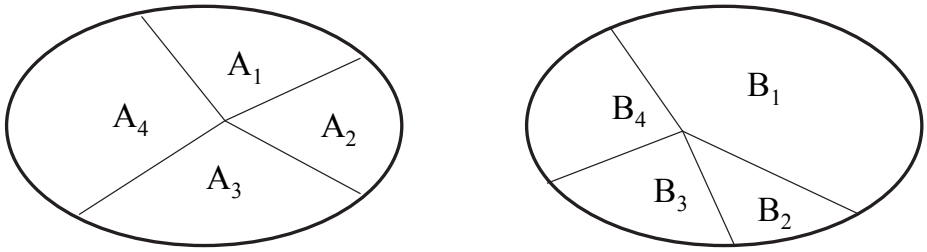


Figure 2.6: Two clusterings $A_1 \ldots A_m$ and $B_1 \ldots B_n$ of the same elements, such that $A_1 \cup A_2 \approx B_1, A_3 \approx B_2 \cup B_3$, and $A_4 \approx B_4$.

The greedy algorithm used to find these groups or *superclusters* can be summarized in the following two steps:

1. take, for each node, the edge with the highest weight,
2. find the connected components in this edge-reduced bi-graph, which define the cluster groupings on each side and their correspondences and connect them in 'superclusters' (see figure 2.7.b).

The idea of merging clusters to form larger clusters has been previously used in Sharan and Shamir (2000) in a refinement step of the CLICK algorithm, described in section 1.8.3, where kernels (clusters) with highest similarity, exceeding a given threshold, are merged to produce a new kernel. Merging clusters is also used as a refinement of the Highly Connected Subgraphs (HCS) algorithm, by Hartuv *et al.* (1999), who represent the data as a similarity graph and define clusters as subgraphs with connectivity above half the number of vertices, because the algorithm tends to be too severe and may split true clusters. However, we are not exploiting the similarities between the clusters in the same clustering as they do; instead, we are using the comparison of two different clusterings.

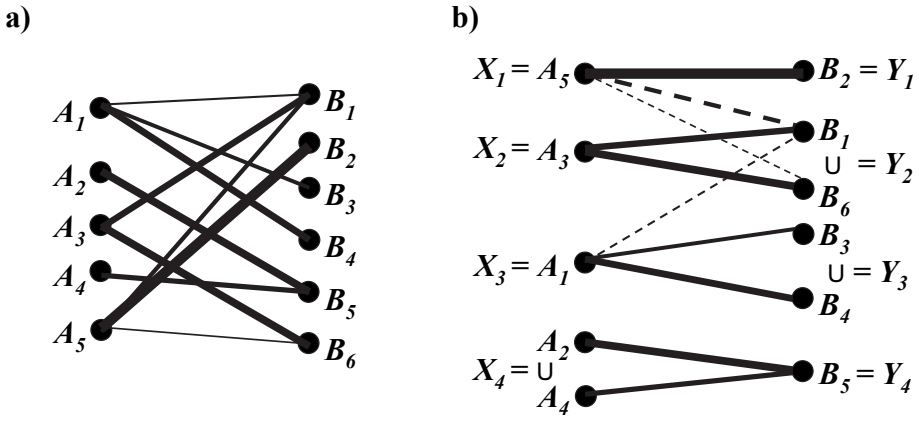


Figure 2.7: Finding the superclusters of graph drawn in a) as connected components, using the most significant edges. Cluster A_5 corresponds to cluster B_2, A_3 corresponds to $B_1 \cup B_6$, A_1 corresponds to $B_3 \cup B_4$ and $A_2 \cup A_4$ corresponds to B_5.

Theoretically it is possible to obtain a fully connected graph with this approach, which leads to the trivial solution $X_1 = Y_1 = G$, but in practice our results on different gene expression datasets show that this does not happen normally. In artificial data sets we obtained fully connected graphs in the following two cases:

a) when there is no underlying structure in the data;

b) when the number of clusters in the clusterings is smaller than the number of underlying groups and the elements are homogeneously distributed among the groups.

This way, we can define heuristically a correspondence between both clusterings **A** and **B** that turns to be optimal in terms of certain scoring functions. In particular, consider the following problem: "*Given two partitions* $\mathbf{A}= \{A_1, \ldots, A_m\}$ *and* $\mathbf{B}= \{B_1, \ldots, B_n\}$ *of the same data set* $G = \{g_1, ..., g_N\}$*, where* $g_k, (k = 1, ..., N)$ *are* d*-dimensional vectors* $(g_{k1}, ..., g_{kd})$*, obtained by some clustering procedure, we want to find:*

a) *the two best partitions* $\boldsymbol{A}_p = \{A_{1,p}, \ldots, A_{p,p}\}$ *and* $\boldsymbol{B}_p = \{B_{1,p}, \ldots, B_{p,p}\}$*, where* $p \leq \min\{m, n\}$*, less fine than the original ones, according to an objective function*

$f(\boldsymbol{A}_{\mathrm{p}}, \boldsymbol{B}_{\mathrm{p}})$, *and*

b) *a one-to-one mapping between the sets of these new partitions,* $\mathrm{g} : \boldsymbol{A}_{\mathrm{p}} \mapsto \boldsymbol{B}_{\mathrm{p}}$."

The condition of $\mathbf{A}_{\mathrm{p}}$ being less fine than $\mathbf{A}$ means that for all $\mathrm{A_i} \in \mathbf{A}$ there exists only one $\mathrm{A_{j,p}} \in \mathbf{A}_{\mathrm{p}}$ such that $\mathrm{A_i} \subset \mathrm{A_{j,p}}$. We are interested in minimizing the probability that the overlap between the sets $\mathrm{A_{j,p}}$ and $\mathrm{B_{j,p}}$, matched by the mapping, has occurred by chance. Therefore, a possible objective function could be:

$$f(\cdot,\cdot) = \sum_{\mathrm{k=1}}^{\mathrm{p}} \mathrm{P}(\mathrm{A_{k,p}} \cap \mathrm{B_{k,p}}) \mid \text{hypergeometric distribution}),$$

that is, the sum of the probabilities of obtaining these intersection sizes, assuming an underlying hypergeometric distribution. However, this may lead to undesirable results, as finding large intersections could be as unlikely as finding small ones. Therefore, intersections sizes should be taken into account in the objective function, like in:

$$f_1(\cdot,\cdot) = \sum_{\mathrm{k=1}}^{\mathrm{p}} \frac{1}{\mid \mathrm{A_{k,p}} \cap \mathrm{B_{k,p}} \mid +\mathrm{a}} \times \mathrm{P}(\mathrm{A_{k,p}} \cap \mathrm{B_{k,p}} \mid \text{hypergeometric distribution}),$$

where

$$a = \begin{cases} 0, & \text{if } \mid \mathrm{A_{k,p}} \cap \mathrm{B_{k,p}} \mid > 0 \\ \epsilon > 0, & \text{if } \mid \mathrm{A_{k,p}} \cap \mathrm{B_{k,p}} \mid = 0 \end{cases}$$

avoids zeros in the denominator.

The most obvious algorithm to find the two best partitions and the corresponding one-to-one mapping is to compute by "brute force" all possible pairs of partitions and ways of mapping them and output any mapped pair with minimum value of f. But the number of possible mappings grows exponentially and this method turns to be non-feasible.

Proposition 2.1 *The number of possible one-to-one mappings between the sets of clusters* $\boldsymbol{A}$ *and* $\boldsymbol{B}$*, with* m *and* n *clusters respectively, is given by*

$$M = \sum_{q=1}^{min\{m,n\}} R_q^{(m} \cdot S_q^{(n} \cdot q!$$

where $R_q^{(m}$ *and* $S_q^{(n}$ *are computed recursively as:*

$$\begin{aligned} R_1^{(m} &= 1 \\ R_q^{(m} &= \frac{q^{m-1}}{(q-1)!} - \sum_{p=1}^{q-1} \frac{R_p}{(q-p)!}, \qquad q = 2, \ldots, m \\ S_1^{(n} &= 1 \\ S_q^{(n} &= \frac{q^{n-1}}{(q-1)!} - \sum_{p=1}^{q-1} \frac{S_p}{(q-p)!}, \qquad q = 2, \ldots, n \end{aligned}$$

Proof: We can compute the number of possible comparisons by computing: on one hand, the number of ways in which a set of m clusters can be merged into $1, 2, \ldots, p$ superclusters, on the other hand, the number of ways in which a set of n clusters can be merged into $1, 2, \ldots, p$ superclusters, and finally, the number of ways in which k superclusters on either side can be mapped, $k = 1, \ldots, p$.

Consider clustering **A**, with m groups and clustering **B**, with n clusters. With the m groups of **A** we can form 1 supercluster (merging all the clusters), or 2 superclusters, or 3 superclusters,..., or m superclusters (m singletons, where we are treating clusters as elements). Represent each cluster by a numbered ball and suppose that we assign to each cluster a label that represents the number of an urn where we can "introduce" the cluster. If there are q urns, there are $\mathbb{VR}_q^m = q^m$ ways to assign the balls to the urns, where $\mathbb{VR}_q^m$ denotes the total number of variations with repetition of q elements choose m. However, some of the distributions are equivalent and we should take into account only a representative for each distribution. For example, if we have two urns, we can assign all the balls to the first urn or to the second urn, but we are interested in counting only one of these possibilities. For each number of urns $p = 1, \ldots, m$, let $R_p^{(m}$ be the number of representatives of the possible distributions of m numbered balls in p urns.

For a fixed number q of urns there are

$$\mathbb{C}_q^p \cdot \mathbb{P}_p = \binom{q}{p} \cdot p! = \frac{q!}{(q-p)!} = \mathbb{V}_q^p$$

different distributions equivalent to the same representative $R_p^{(m}$, $p = 1, \ldots, q$, where $\mathbb{C}_q^p$ is the combinatorial number $\binom{p}{q}$, $\mathbb{P}_p$ is the number of permutations of p elements and $\mathbb{V}_q^p$ is the number of variations of q elements choose p.

Therefore, for each $q = 1, \ldots, m$ we can establish that

$$\mathbb{VR}_q^m = q^m = \sum_{p=1}^{q} R_p^{(m} \cdot \mathbb{C}_q^p \cdot \mathbb{P}_p = \sum_{p=1}^{q} R_p^{(m} \frac{q!}{(q-p)!} = q! \sum_{p=1}^{q} \frac{R_p^{(m}}{(q-p)!}$$

and therefore

$$\begin{aligned} R_1^{(m} &= 1 \\ R_q^{(m} &= \frac{q^{m-1}}{(q-1)!} - \sum_{p=1}^{q-1} \frac{R_p^{(m}}{(q-p)!}, \qquad q = 2, \ldots, m. \end{aligned} \tag{2.2.1}$$

From these m equations, we can determine the m values $R_1^{(m}, \ldots, R_m^{(m}$ and conclude that we can assign the m groups to 1 urn in $R_1^{(m} = 1$ way, to 2 indistinguishable urns in $R_2^{(m}$ ways, and so forth.

Equivalently, for clustering $\mathbf{B}$, where there are n clusters, we can determine the number of representatives $S_1^{(n}, \ldots, S_n^{(n}$ of the distribution of n balls in $q = 1, \ldots, n$ urns, with the n equations

$$\begin{aligned} S_1^{(n} &= 1 \\ S_q^{(n} &= \frac{q^{n-1}}{(q-1)!} - \sum_{p=1}^{q-1} \frac{S_p^{(n}}{(q-p)!}, \qquad q = 2, \ldots, n. \end{aligned} \tag{2.2.2}$$

Finally, as the order in the assignments of groups from either side does matter, the total number of one-to-one mappings M that can be established, can be computed as

$$M = \sum_{p=1}^{min\{m,n\}} R_p^{(m} \cdot S_p^{(n} \cdot \mathbb{P}_p = \sum_{p=1}^{min\{m,n\}} R_p^{(m} \cdot S_p^{(n} \cdot p!. \tag{2.2.3}$$

■

Due to the exponential growth of the number of possible comparisons, we only ran the greedy algorithm for data sets where we performed k-means with a relatively small number of clusters. We tested that it returns the optimal solution to this problem, in terms of the objective function f_1.

2.2.2. Results

To check the performance of the greedy algorithm, we ran k-means twice on each generated data set and computed the average percentage of misclassified elements in each cluster (with the convention of considering each cluster as part of the true group that is the most voted by its elements) and the average number of superclusters.

Example 2.3 *Model 1 of size 250*

We generated 100 data sets from model 1, considering cluster sizes randomly chosen between 1 and 250, but also considered 100 data sets of fixed size (250 observations for each cluster). In table 2.3 we show the summary of the results for data sets with 4 and 7 true groups. The two flat clusterings had k=8 and k=11 clusters. The noise level ranges between 1 and 6.

In the case of 4 real groups, the greedy algorithm shows the worst performance when the data is highly noisy ($\sigma = 6$), increasing the percentage of misclassified elements by less than 2 (from 18.96 % in k-means, with k=11, to 20.79 % after grouping). However, the average number of superclusters found is often less than 7, versus the 8 and 11 clusters of k-means. Similarly, in the case of 7 real groups the increase in the percentage of misclassified elements is not larger than 3, except for the case of $\sigma = 1, 2, 3$ and k=11, where the merging causes this percentage to increase in 9. However, this is a consequence of the error rate obtained with k-means when k=8. The average number of superclusters is close to 7 in all the data sets. ■

Example 2.4 *Model 2*

	♯ true groups	Noise B/W	% elements closer to other centres	Clustering 1 % of misclassification		Clustering 2 % of misclassification		Average number of superclust
				k-means (k=8)	Grouping	k-means (k=11)	Grouping	
	4	6.09	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	6.37
Same size	4	3.00	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	6.35
clusters	4	2.03	0.20 %	2.77 %	2.77 %	3.59 %	3.59 %	6.70
#genes	4	1.47	2.60 %	4.14 %	4.30 %	4.73 %	5.01 %	6.88
=1000	4	1.22	8.40 %	11.44 %	12.19 %	12.25 %	13.25 %	6.81
	4	0.99	16.70 %	23.01 %	23.72 %	24.16 %	25.56 %	7.03
	4	7.56	0.00 %	0.80 %	0.89 %	0.09 %	0.89 %	6.42
Different	4	3.80	0.70 %	1.38 %	1.95 %	1.32 %	1.96 %	6.44
size	4	2.48	2.97 %	4.38 %	4.74 %	3.73 %	4.47 %	6.50
clusters	4	1.93	4.55 %	6.52 %	6.65 %	6.39 %	6.72 %	6.92
	4	1.55	8.57 %	10.36 %	10.92 %	10.59 %	11.64 %	7.11
	4	1.23	14.69 %	18.49 %	19.73 %	18.96 %	20.79 %	6.96
	7	7.15	0.00 %	10.14 %	14.28 %	5.00 %	14.28 %	6.54
Same size	7	3.59	0.11 %	9.53 %	11.94 %	3.40 %	11.94 %	6.74
clusters	7	2.40	1.60 %	9.03 %	11.30 %	4.69 %	11.40 %	6.84
#genes	7	1.80	6.51 %	10.30 %	10.69 %	8.54 %	11.23 %	7.03
=1750	7	1.41	2.80 %	15.60 %	16.00 %	16.03 %	17.43 %	7.27
	7	1.22	17.60 %	21.95 %	22.35 %	24.41 %	25.71 %	7.59
	7	7.17	0.00 %	3.42 %	4.51 %	1.75 %	4.51 %	6.70
Different	7	3.65	4.40 %	4.59 %	5.41 %	1.92 %	5.44 %	6.70
size	7	2.42	2.56 %	4.91 %	5.19 %	3.07 %	5.69 %	6.88
clusters	7	1.84	8.34 %	10.39 %	10.61 %	9.77 %	10.70 %	5.07
	7	1.51	16.15 %	14.75 %	15.32 %	15.17 %	16.29 %	7.19
	7	1.32	17.90 %	19.55 %	19.96 %	19.73 %	20.86 %	7.33

Table 2.3: The measure of the noise B/W is the ratio between the average distance B between clusters centres and the average distance W between each point and the centre of the cluster where it belongs.

Next, we generated 100 data sets from Model 2, considering noise levels ranging between 1 and 6. We ran k-means with k = 5 and k = 6.

The average number of superclusters is close to 4 (only one group more than the real number of groups) and, again, merging clusters does not increase the percentage of misclassified elements considerably. Even though these percentages reach values of 40 %, it has to be noticed

Noise B/W	% elements closer to other centres	Clustering 1 % of misclassification		Clustering 2 % of misclassification		Average number of superclust
		k-means (k=5)	Grouping	k-means (k=6)	Grouping	
5.46	0.53 %	1.21 %	1.70 %	1.21 %	1.81 %	4.08
2.78	10.22 %	13.18 %	15.65 %	13.98 %	16.61 %	4.32
1.84	22.23 %	24.43 %	26.30 %	24.43 %	26.52 %	4.46
1.40	30.42 %	30.57 %	33.10 %	30.57 %	33.30 %	4.50
1.12	36.79 %	36.08 %	37.58 %	36.08 %	37.92 %	4.42
0.96	40.43 %	38.10 %	40.19 %	38.10 %	40.33 %	4.36

Table 2.4: Greedy algorithm run on 100 data sets from model 2, with noise levels $\sigma = 1, \ldots, 6$.

that the proportion of elements that are closer to other centres than their own is very close to these percentage, and these values cannot be considered a failure of the method. ■

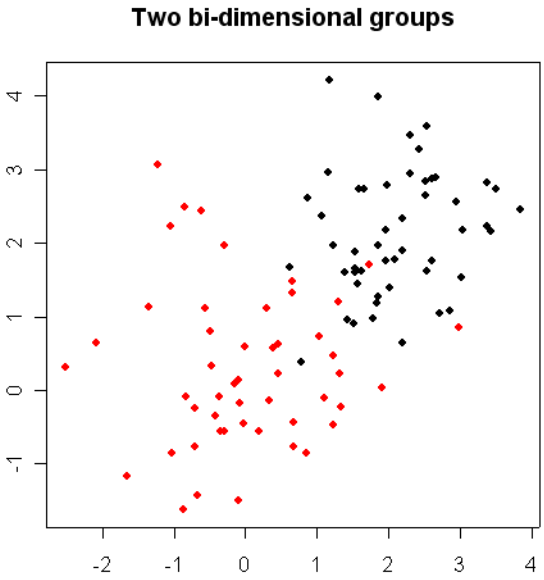
2.3. Comparison of a hierarchical and a flat clustering

An immediate approach to face the comparison of a flat and a hierarchical clustering is to cut the dendrogram, so that a set of flat clusterings is obtained, and then we can apply the algorithms described in section 2.2. However, it is not always easy or trivial to select the points at which the dendrogram should be cut off to produce flat clusters. Often the cut-off points are selected by visual inspection, based on the previous knowledge that one has about the data, but there are no methods that systematically give the best levels to cut the dendrogram.

Another common practice is to cut different branches of the tree at the same height to produce a given number of flat clusters, but yet this can lead to undesired results. See for example figure 2.8.

We have represented in figure 2.8.a) two bi-dimensional Gaussian groups C_1 and C_2 of size 50, centered at $\mu_1 = (0, 0)$ (red cluster) and $\mu_2 = (2, 2)$ (black cluster), with covariance matrices

a)



b)

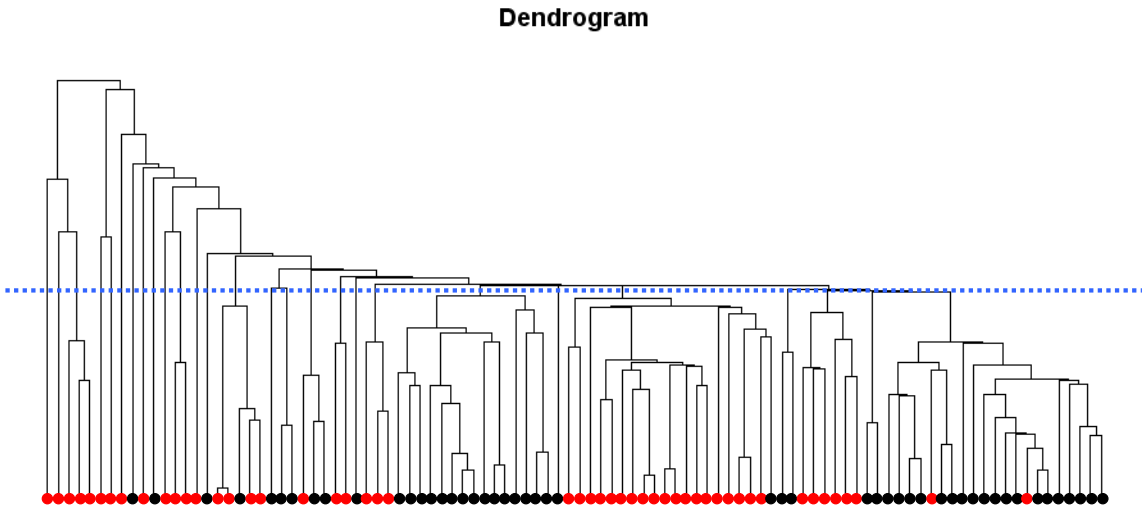


Figure 2.8: a) Two bidimensional Gaussian groups centered at (0,0) and (2,2), with 50 elements each. b) Dendrogram produced by the single linkage hierarchical clustering. Memberships of the elements are shown with red and black dots. The dashed blue lines shows the height at which cutting the tree produces 25 clusters.

$\Sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $\Sigma_2 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$. If we run single linkage hierarchical clustering on

these data we obtain the dendrogram shown in figure 2.8.b. The memberships of the leaves are displayed as red and black dots, respectively. To roughly identify the two groups cutting the tree at a fixed height we need to produce around 25 clusters (this height is shown as a dotted blue line). Notice that if we prune the tree to get 10 clusters, we only detach 13 elements in C_1 from a large cluster containing 85 elements; 20 clusters detach 24 elements in C_1 from a cluster of size 67 and the 25 clusters contain 14 singletons, while the largest cluster, of size 29, contains 8 elements from C_1 and 21 from C_2. A simple data set like the one here described needs a more sophisticated way to cut the dendrogram.

We have implemented a method that cuts different branches of the hierarchical tree at "optimal" levels, which may be different at different branches, to find the best matches with a given flat clustering. The method we use explores the tree depth-first, starting from the root. The basic idea is the following: suppose we are at branch A. Assume that the graph is visualized so that the hierarchical tree is on the left, while the flat clustering is on the right (see Figure 2.9).

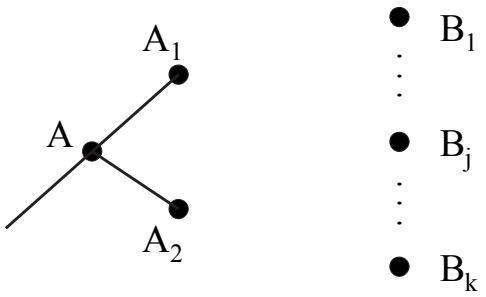


Figure 2.9: A_1 and A_2, descendants of a sub-branch A of the original tree, are tested as potential split points against the flat clustering $B_1, \ldots, B_k$.

We compute a certain score $\sigma = S(A)$, defined later (see sections 2.3.1 and 2.3.2). Then we run one iteration of the gravity-centre algorithm (section 2.2) on the right side for the children nodes A_1 and A_2, and compute the score $\sigma_1 = S'(A_1, A_2)$. Next, we swap the nodes A_1 and A_2, run the algorithm, and compute $\sigma_2 = S'(A_2, A_1)$. If either σ_1 or σ_2 is higher than σ, we split

the tree using the ordering giving the higher score. If both σ_1 and σ_2 are lower than σ, then we do not split A and start exploring the corresponding (adjacent) node to decide whether or not we split it.

```
procedure correspondence(A,Y); Y global;

    if A is a leaf
     then
       link(A, Y)
     else
       find the two descendants of A : A1 and A2
       compute σ = S(A), σ1 = S'(A1, A2) and σ2 = S'(A2, A1)
       Y12 = gravity_center((A1, A2), Y)
       Y21 = gravity_center((A2, A1), Y)
       if max(σ1, σ2) < σ
        then
         link(A, Y)
        else if (σ1 ≥ σ2)
           then
             Y = Y12
             link( (A1, A2), Y)
            correspondence(A1, Y)
            correspondence(A2, Y)
           else
             Y = Y21
            link( (A2, A1), Y)
            correspondence(A2, Y)
            correspondence(A1, Y)
```

Box 2.3.1. The procedure *gravity_center*() finds the optimal ordering of the nodes on the flat side according to the gravity-center algorithm and the procedure *link*() adds to the bi-graph the edges that connect the set of nodes in the first argument with the nodes in the second one, in the ordering given by Y.

The *correspondence* algorithm is described in Box 2.3.1 more formally through its pseudo-

code. Here, A is a tree with a fixed layout (or a node in a special case), and $Y = (y(B_1), \ldots, y(B_k))$ is the set of ordinates of the flat clusters.

An improved version of this algorithm, including a "look-ahead" step, is given in section 2.3.3. In next sections we describe the two different scoring functions that we use for S. The first one, S_1, is based on the concept of mutual information – we try to find a split of the tree optimizing the mutual information between the two clusterings; the second one, S_2, is based on graph layout aesthetics.

2.3.1. Information Theoretic-based scoring function

The concepts relative to the Minimum Description Length principle (see section 1.7) have been previously used in clustering problems, for example, in Hansen and Yu (2001) and Jörnsten and Yu (2003). Hansen and Yu (2001) review the MDL principle and illustrate its use in clustering analysis, encoding the number of clusters, the number of points belonging to each cluster, the parameters needed to specify each cluster model, the cluster membership for each data point, and finally the data are encoded using the distribution of the specified model. Jörnsten and Yu (2003) propose a method for the simultaneous clustering of genes and the subset selection of gene clusters for sample classification, providing an MDL code length for both genes and sample class labels.

In our approach, intended to compare a hierarchical and a flat clustering, the underlying idea is the following. Consider that there are two people, a message sender and a message receiver, each knowing a particular clustering of the same data set. If the sender wants to describe his/her clustering, given that he/she knows the clustering of the receiver, and both clusterings are equal, then a short message he/she could send to the receiver is "Both clusterings are the same". If both clusterings are slightly different, he/she could send a message like: "Both clusterings are the same, except for elements A, B, C and D". However, if both clusterings are

quite different, the message needed to describe the sender's clustering would be much larger, possibly detailing all the elements in the data set.

In this way, we are interested in finding the length of a code that allows the sender to describe his/her clustering with the minimum possible number of bits, provided that the other clustering is known to him/her. It is also important to notice that none of the clusterings that are compared in our approach is more important than the other, therefore we have to consider the length of a message "sent" from clustering $\mathbf{A}$, knowing $\mathbf{B}$, and also a message "sent" from $\mathbf{B}$, knowing $\mathbf{A}$.

More precisely, suppose the flat clustering is $\mathbf{B} = \{B_1, \ldots, B_n\}$. If the dendrogram has m leaves $\mathbf{A} = \{A_1, \ldots, A_m\}$, we can express the distribution of genes across the dendrogram as $p_i^A = \frac{|A_i|}{N}$ (where N is the number of elements in $G = A_1 \cup \ldots \cup A_m$). The distribution of genes in a particular leaf A_i across the flat clusters $\{B_1 \ldots B_n\}$ will be

$$p_{ij}^A = \frac{|A_i \cap B_j|}{|A_i|},\ j = 1 \ldots n.$$

Notice that p_{ij}^A is the conditional probability of finding an arbitrary gene from A_i in cluster B_j. Therefore, the conditional Shannon's entropy (Shannon, 1948) of $\mathbf{B}$, given $\mathbf{A}$, will be

$$H(\mathbf{B}|\mathbf{A}) = \sum_{i=1}^{m} p_i^A H(\mathbf{B}|A_i) = -\sum_{i=1}^{m} p_i^A \sum_{j=1}^{n} p_{ij}^A \log_2 p_{ij}^A = \\ = -\frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{n} |A_i \cap B_j| \log_2 \frac{|A_i \cap B_j|}{|A_i|} \tag{2.3.4}$$

Note that $-\sum_{i=1}^{m} \sum_{j=1}^{n} \log_2 \frac{|A_i \cap B_j|}{|A_i|}$ is the number of bits needed to encode the clusters in clustering $\mathbf{B}$, knowing $\mathbf{A}$.

Similarly, denoting the probabilities $p_j^B = \frac{|B_j|}{N}$, and $p_{ij}^B = \frac{|A_i \cap B_j|}{|B_j|}$, the Shannon's entropy of $\mathbf{A}$ given the flat clustering $\mathbf{B}$ is

$$H(\mathbf{A}|\mathbf{B}) = \sum_{j=1}^{n} p_j^B H(\mathbf{A}|B_j) = -\sum_{j=1}^{n} p_j^B \sum_{i=1}^{m} p_{ij}^B \log_2 p_{ij}^B = \\ = -\frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{n} |A_i \cap B_j| \log_2 \frac{|A_i \cap B_j|}{|B_j|} \tag{2.3.5}$$

According to the MDL principle we try to minimize the length of the message that needs to be sent to describe the clustering $\mathbf{B}$ if the clustering $\mathbf{A}$ is known to the receiver, or vice versa. These message lengths are $H(\mathbf{B}|\mathbf{A})$ and $H(\mathbf{A}|\mathbf{B})$, respectively. However, we also need to add to the message the coding of the clusters themselves, which adds to the length in bits a term given by the expressions

$$-\frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{n} \log_2 p_{ij}^A$$

and

$$-\frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{n} \log_2 p_{ij}^B,$$

respectively. To find a compromise between the information about the dendrogram conveyed by knowing the flat clustering and vice versa, we define a symmetric score given by the average of $H(\mathbf{B}|\mathbf{A})$ and $H(\mathbf{A}|\mathbf{B})$ plus the length of the coding of both clusterings

$$S_1(\mathbf{A},\mathbf{B}) = -\frac{1}{2}\left(H(\mathbf{B}|\mathbf{A}) + H(\mathbf{A}|\mathbf{B}) - \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{n} \log_2 p_{ij}^A p_{ij}^B \right). \tag{2.3.6}$$

Suppose we want to decide whether or not to split branch A_r. We denote its descendants by A_r^* and A_{r+1}^*. We will split A_r if the average of the length of the messages that encode the information about one clustering contained in the other decreases when replacing the branch with its descendants. Since splitting a branch A_i affects only the i-th term of the entropy expression, we define the scoring function for a given branch A_r as

$$\sigma = S_1(A_r) = \frac{1}{2N}\sum_j (|A_r \cap B_j| + 1)\log_2 p_{rj}^A + \frac{1}{2N}\sum_j (|A_r \cap B_j| + 1)\log_2 p_{rj}^B \tag{2.3.7}$$

and the score for its descendants as $\sigma_1 = \mathrm{S}_1(\mathrm{A}^*_\mathrm{r}) + \mathrm{S}_1(\mathrm{A}^*_{\mathrm{r}+1})$.

As the scoring σ_1 thus defined is symmetric in its arguments, if $\sigma_1 > \sigma$ the ordering given in the tree will be the one with a smaller number of edge crossings. For the look-ahead algorithm we generalize this score from two leaves to an arbitrary subtree.

A similar idea of measuring the gain or loss of information is described in Jonnalagadda and Srinivasan (2004), where they perform k-means clustering successively on the same data set, increasing in each iteration the number of clusters by one. However in our approach we use this measure to find the best correspondence between two different clusterings.

To understand how this scoring works we can follow this approach. Consider a first graph consisting of one node A in the hierarchical side, with c elements, that is connected to the two flat clusters B_1, with d elements, and B_2, with $c - d$ elements (see figure 2.10). Next, consider a second graph, obtained from the first one by splitting node A into A_1, with x elements, and A_2, having $c - x$ elements. If A_1 and B_1 have a common elements, then A_1 and B_2 have $x - a$ elements in common, A_2 and B_1 share $d - a$ elements and A_2 and B_2 have $c - d - x + a$ common elements.

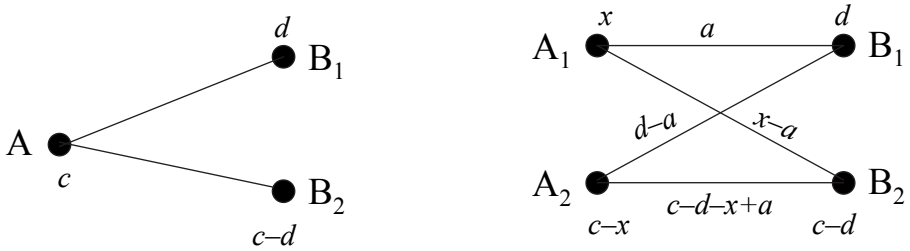


Figure 2.10: Distribution of the c elements of node A when splitting it to obtain nodes A_1 (with x elements) and A_2 (with $c - x$ elements).

Then compute the scores $\sigma = \mathrm{S}_1(\mathrm{A})$ and $\sigma_1 = \mathrm{S}_1(\mathrm{A}_1) + \mathrm{S}_1(\mathrm{A}_2)$; for fixed values of a we can make a bi-dimensional plot showing the sign of $\sigma_1 - \sigma$, with axis x and d. If the difference is positive, we plot a red dot; otherwise, we plot a blue dot. In figure 2.12 we show these plots

for $a = 1$, 10, 20 and 50 for an initial node with $c = 150$ elements. As an example, the values $a = 1$, $x = 60$ and $d = 50$ yields a blue dot, indicating that node A should not be split. In fact, the resulting split graph, shown in figure 2.11.a) shows that this split should not take place. Instead, for values $a = 1$, $x = 100$ and $d = 50$ we get a red dot, corresponding to the graph shown in figure 2.11.b), and obviously this split should be accepted, as it is able to identify to distinct groups.

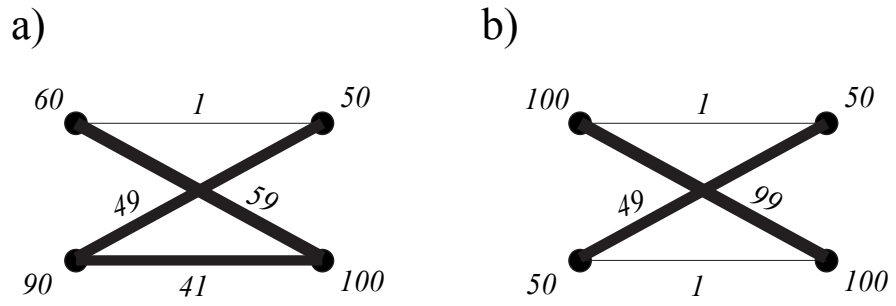


Figure 2.11: Split graphs for a) $a = 1$, $x = 60$ and $d = 50$; b) $a = 1$, $x = 100$ and $d = 50$. The edge thickness represents the number of elements common to both nodes.

2.3.2. Graph layout aesthetics-based scoring function

The second score that we use, S_2, is based on a graph layout aesthetics demonstrated in Figure 2.13: if splitting a node A into A_1 and A_2 creates 'thick' connections from each of the A_i to each of the B_j (as in Figure 2.13, top), but not reciprocally (as in Figure 2.13, middle), then the splitting is rewarded, otherwise it is penalized. The intermediate cases depend on the exact weights – intuitively they are difficult to decide.

To achieve this, for branch A we compute the product

$$S_2(A) = \sum_j \frac{|A \cap B_j|}{|A \cup B_j|} \sum_j |A \cap B_j|^2. \tag{2.3.8}$$

The rationale of this scoring can be explained as follows. The first term corresponds to the sum of Jaccard indices (Harper, 1999), $\frac{|A \cap B_j|}{|A \cup B_j|}$, and rewards the cases where most of the

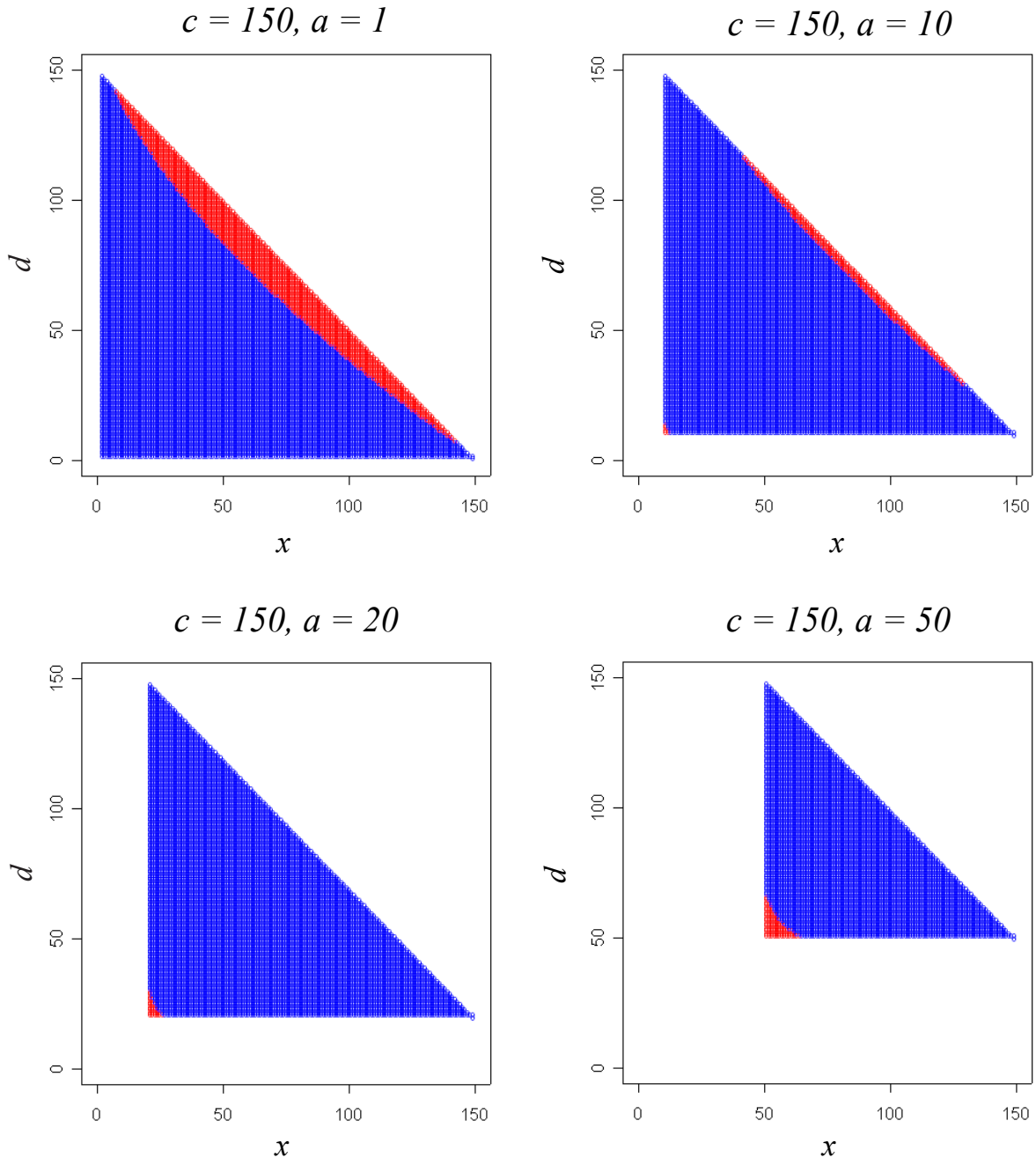


Figure 2.12: Bi-dimensional plots of the sign of $\sigma_1 - \sigma = \mathrm{S}_1(\mathrm{A}_1) + \mathrm{S}_1\mathrm{A}_2 - \mathrm{S}_1(\mathrm{A})$, depending on the values of x, d and a $(= 1, 10, 20, 50)$ for a node A with $c = 150$ elements.

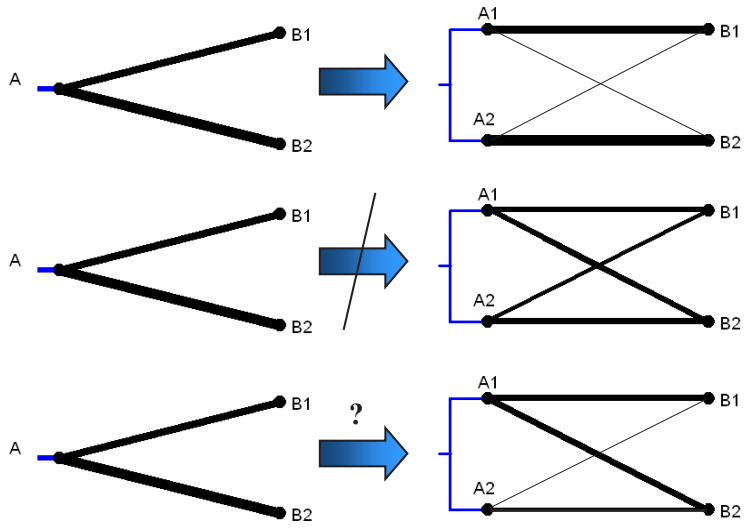


Figure 2.13: The scoring function S_2 allows to split when corresponding clusters are identified (top) and prevents unnecessary splits (middle). Non-symmetric splits (bottom) are allowed depending on the thickness of the corresponding edges.

elements are present in corresponding clusters on both sides, regardless of the absolute size. The second term is the addition of the square of the number of elements in common $|A \cap B_j|^2$, and rewards fewer thicker edges (i.e., edges representing larger absolute number of common elements), penalizing a relatively larger number of smaller edges.

For the descendants A_1 and A_2 we compute σ_1 as

$$S'_2(A_1, A_2) = \sum_j \left(\frac{|A_1 \cap B_j|}{|A_1 \cup B_j|} + \frac{|A_2 \cap B_j|}{|A_2 \cup B_j|} \right) \times \sum_j \left(|A_1 \cap B_j|^2 + |A_2 \cap B_j|^2 \right) - \text{crossings}(A_1, A_2), \tag{2.3.9}$$

where $\text{crossings}(A_1, A_2)$ is the number of edge crossings within the subtree that results from expanding branch A. This number of crossings is based on the weights: it is computed as

$$\sum_{j>1} \sum_{k<j} w(A_1, B_j) \times w(A_2, B_k)$$

which is not symmetric in its arguments.

Similarly to the information theoretic based score, we can draw a bi-dimensional plot for the sign of $S'_2(A_1, A_2) - S_2(A)$, computed for the graphs displayed in figure 2.10, to help understand how the scoring function works. Figure 2.14 shows these plots for the same values of c and a used in figure 2.12. Notice that these plots are different from the ones obtained when plotting $S'_2(A_2, A_1) - S_2(A)$. This asymmetry can be observed by comparing figures 2.14 and 2.15.

We can see that this second score is less restrictive than the information theoretic based one, as all red zones in these new plots include the red zones in the previous plots. For example, the parameters $a = 1$, $x = 30$ and $d = 100$ yield a red dot in both figures 2.14 and 2.15, while we have a blue dot for the information theoretic based score, and therefore, we will split the graph or not depending on whether we use score S_2 or S_1. The parameters $a = 1$, $x = 50$ and $d = 70$ correspond to blue dots for $S_1(A_1, A_2) - S_1(A)$ and $S'_2(A_1, A_2) - S_2(A)$, but to a red dot for $S'_2(A_2, A_1) - S_2(A)$, and again we will split the graph if we use score S_2.

The computational experiments described in sections 2.4.1 and 2.4.2 show that this heuristic scoring function performs almost as well as the information-theoretic one, outperforming it for a small number of look-ahead steps. After determining how the branches in the dendrogram will be collapsed, we can run the greedy algorithm on the corresponding flat clusterings. As we will show, the resulting number of superclusters is usually close to the number of real groups in the data, which can be partially or fully restored.

2.3.3. The look-ahead algorithm

The look ahead algorithm adds to the algorithm described previously the possibility of checking how the selected score S behaves some nodes ahead, after finding a node whose split is not recommended by S, and then deciding whether or not the split will take place.

More precisely, if we are exploring node A, whose descendants are A_1 and A_2, and we compute the scores σ, σ_1 and σ_2, and it turns out that both σ_1 and σ_2 are lower than σ, we start a look-ahead algorithm to explore the tree depth-first starting from A, looking h steps

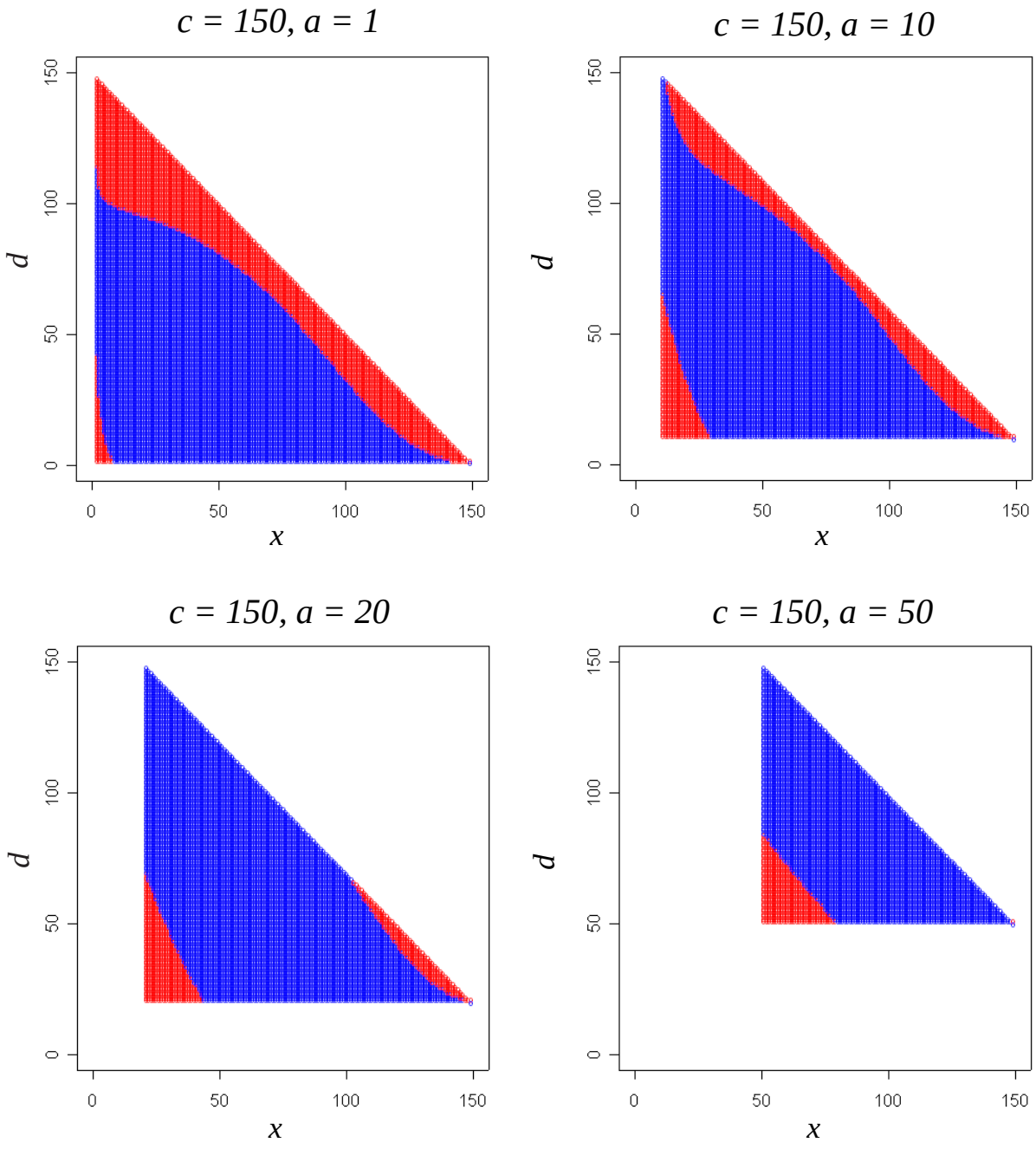


Figure 2.14: Bi-dimensional plots of the sign of $S_2(A_1, A_2) - S_2(A)$, depending on the values of x, d and a $(= 1, 10, 20, 50)$ for a node A with $c = 150$ elements.

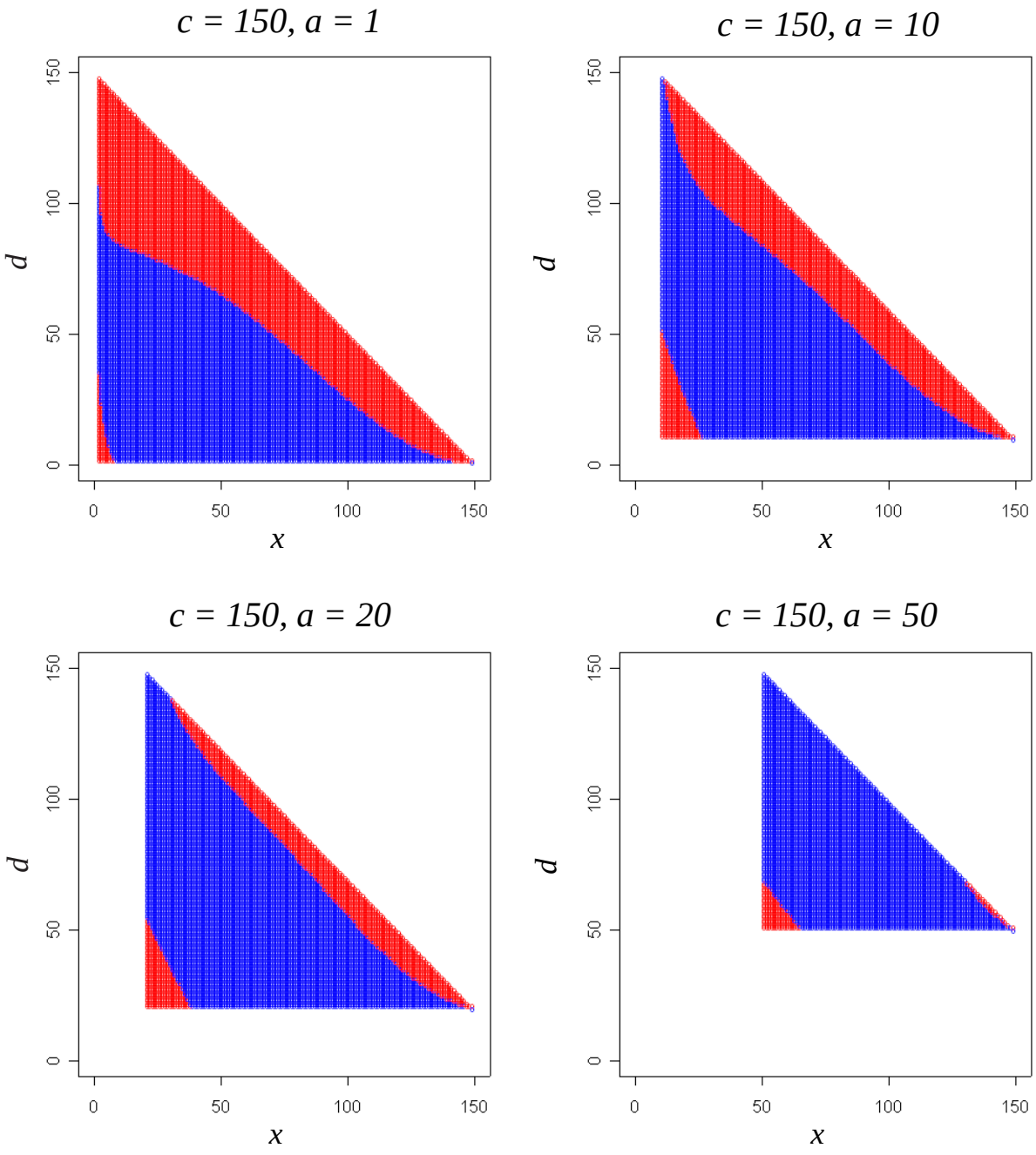


Figure 2.15: Bi-dimensional plots of the sign of $S_2(A_2, A_1) - S_2(A)$, depending on the values of x, d and a $(= 1, 10, 20, 50)$ for a node A with $c = 150$ elements.

ahead, where h is given by the user. If, after h steps, none of the scores has improved, we do not split the node A, otherwise we split the tree to obtain the path to the node with the highest score. This is illustrated in Figure 2.16: when we split the branch marked with a red dot in graph a) we get a value of the score in graph b) that is worse than the one we had without splitting. However, if we look ahead one step further we manage to identify two distinct clusters, corresponding to a value of the score in graph c) that is better than that of a).

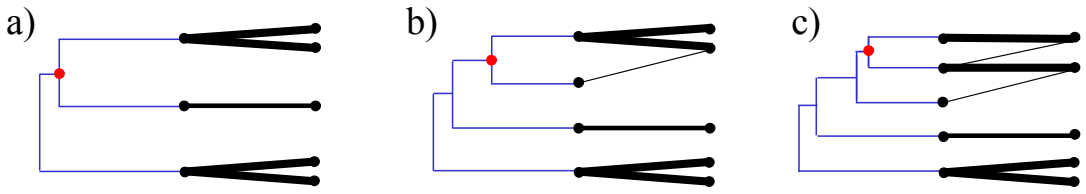


Figure 2.16: Performance of the look-ahead algorithm: a decrease in the value of the score in *b*) with respect to that of *a*) is overcome with a better value in *c*).

When a node A is being examined to decide whether or not we are going to split it into A_1 and A_2 we set a pointer or finger at A to remember where we will come back if after exploring h steps ahead there is no improvement of σ. We will denote by σ_1 and σ_2 the scores generalized to the subtree enclosed between the pointer and the two new nodes that are being checked. We have then the following four cases:

a) any of the scores σ_1 or σ_2 improves the value of σ and the new nodes are the children of A, A_1 and A_2; then we split the node A using the ordering giving the higher score. This case corresponds to the splitting described in the simple version of the algorithm. We set the pointer at the next node to be explored (either A_1 or A_2, because we are exploring the tree depth-first).

b) any of the scores σ_1 or σ_2 improves the value of σ and the new nodes, which are at most h+1 steps apart from A, are not the children of A; then we split the node A

using the ordering giving the higher score, because we have managed to improve σ within the h prefixed steps. We set the pointer at the next node to be explored.

c) none of the scores σ_1 or σ_2 improves the value of σ but the new nodes are at most h steps apart from A. Then we add the new nodes to the list of nodes that have to be checked and continue to look ahead.

d) none of the scores σ_1 or σ_2 improves the value of σ and the new nodes are h+1 steps apart from A. Then we go back one step and jump to the next node to be checked. If this node is not a **descendant** of A, then we do not split A and continue exploring the new node.

Notice that these four cases are not considering the possibility of finding a leaf within the h steps. In that case, we simply jump to the next node to be explored. If this is not a **descendant** of A, then we do not split A.

The pseudo-code of the *look-ahead* algorithm is given in the Appendix.

2.4. Results

2.4.1. Results in simulated data

Example 2.5 *Model 1 with cluster size 100*

For comparing a flat and a hierarchical clustering, we ran k-means with k=10, 100 times on the same data set, constructed under Model 1, with cluster sizes randomly chosen between 1 and 100, and considering noise levels ranging between 1 and 6. Since the hierarchical clustering is deterministic we ran it only once, selecting Euclidean distance and the average linkage method. Usually, regardless of the scoring function that we use, the algorithm collapses the dendrogram in such a way that after forming the superclusters there are almost no differences between the

number of misclassified elements in the superclusters and in the original flat clusters.

Firstly, we used the information theoretical-based scoring function, S_1, using several values of the look-ahead parameter h. We ran k-means with k larger than the real number of groups and computed the average number of superclusters, as well as the average number of branches in the collapsed tree.

With no look-ahead, the number of superclusters underestimates the number of true clusters, and the percentage of misclassified elements, even with moderate levels of noise, is high. For example, for the case of 4 real groups, with noise level $\sigma = 4$, and running k-means with k= 10, we get more than 20 % of misclassified elements in the clustering that results from cutting the dendrogram, even if the number of elements closer to centres other than their own is less than 3 % (see table 2.5); for 8 real groups we get even worse results: datasets with noise level $\sigma = 3$ have a misclassification percentage larger than 15 %, while for $\sigma = 4$ is around 35 %. The average number of branches in the tree is always smaller than the real number of groups.

However, looking just one step ahead makes the average number of superclusters come close to the real number of groups and the percentage of misclassification decrease. For example, for 4 true groups, and $\sigma = 4$, the error rate decreases in more than a 35 %, while for 8 real groups and $\sigma = 4$, it decreases in more than a 40 % (see tables 2.5 and 2.6). The average number of branches is always larger than the number of true clusters. For further look-ahead steps, the error rates do not decrease significantly.

Secondly, we used the aesthetics-based score S_2, using the same values of h and k as we used for S_1. With no look-ahead, the restoration of the 4 true groups is accurate: for low levels of noise ($\sigma < 4$) the percentage of misclassified elements is less than 1 % and only for $\sigma = 6$ we get values larger than 15 % (see table 2.7). Moreover, the average number of superclusters is correctly estimated around 4. The number of branches needed to find them ranges between 5 and 9. In the case of 8 real groups the restoration is worse; the number of real groups is underestimated in all the cases, and for $\sigma > 3$ the percentage of misclassification is larger

than 10 %, increasing up to 46 % when $\sigma = 6$. Notice that even if the error rate obtained with k-means is much smaller than that of the hierarchical classification, there is a large increase of the error rate on the flat side when the superclusters are formed, due to the percentage obtained on the hierarchical side. The average number of branches in this case ranges between 6 and 9.

When we look-ahead one step (see table 2.8), the percentage of misclassified elements decreases, while the average number of superclusters increases. With 4 real groups, the differences are not so remarkable as those of 8 real groups, where the decrease in the percentage of misclassification is up to 10. However, the number of superclusters still underestimates the number of true groups. ■

Example 2.6 *Model 2*

We also checked the performance of the method on Model 2, considering noise levels (σ) ranging between 1 and 6, using k-means with k=5. The summary of the results for the information theoretical-based score is given in table 2.9.

Again, not looking ahead any step with this score is not enough to obtain accurate results. In general we obtain less than two superclusters, and an average number of branches in the dendrogram around 2. The percentage of misclassified elements is high, reaching values higher than 40 % for noise levels ≥ 3. However, for h=1, the average number of superclusters is slightly greater than 3, and the average number of branches needed to obtain them oscillates between 8 and 9. The percentages of misclassification decrease respect to those of h = 0. For example, for $\sigma = 2$, the percentage for the hierarchical clustering goes from 23.62 % to 18.25 %, and for $\sigma = 5$, it goes from 48.07 % to 41.23 %. In fact, the obtained percentages are not much higher than the percentage of elements that are closer to other centres than the ones that generated them (see figure 2.5); thus we should not expect values smaller than these for any clustering algorithm.

When we use the aesthetics-based score, we obtain again a better performance when we do

Number of original groups	Noise B/W	% elements closer to other centres than their own	Clustering 1 % of misclassified elements		Clustering 2 % of misclassified elements		Average number of superclusters	Average number of branches in the tree
			Hierarchical tree	Grouping	k-means (k=10)	Grouping		
4	9.17	0.00 %	3.93 %	3.93 %	0.76 %	2.55 %	3.80	3.91
4	4.36	0.00 %	5.72 %	5.72 %	0.69 %	4.45 %	3.66	3.79
4	3.25	0.29 %	9.45 %	9.45 %	0.57 %	4.03 %	3.50	3.66
4	2.15	2.61 %	20.38 %	21.01 %	2.38 %	11.28 %	2.89	3.22
4	1.84	5.26 %	31.86 %	31.92 %	6.04 %	17.68 %	2.23	2.44
4	1.38	6.11 %	50.11 %	50.38 %	10.72 %	19.74 %	1.46	1.62
8	8.95	0.00 %	9.28 %	9.86 %	5.53 %	9.19 %	6.57	5.27
8	4.67	0.06 %	12.48 %	13.35 %	5.91 %	9.50 %	6.33	5.32
8	2.59	0.89 %	14.93 %	15.74 %	3.94 %	10.14 %	6.08	5.20
8	2.18	6.34 %	33.71 %	34.48 %	6.13 %	24.41 %	4.44	4.73
8	1.58	7.17 %	59.78 %	60.34 %	10.61 %	25.41 %	2.27	2.69
8	1.09	13.13 %	75.49 %	75.50 %	20.46 %	28.78 %	1.20	1.27

Table 2.5: Comparison between a hierarchical and a non-hierarchical clustering, with no look-ahead, computing the information theoretical-based score S_1.

81

Number of original groups	Noise B/W	% elements closer to other centres than their own	Clustering 1 % of misclassified elements		Clustering 2 % of misclassified elements		Average number of superclusters	Average number of branches in the tree
			Hierarchical tree	Grouping	k-means (k=10)	Grouping		
4	9.62	0.00 %	1.12 %	1.56 %	0.76 %	0.76 %	3.99	5.40
4	3.98	0.00 %	2.13 %	2.27 %	0.91 %	0.91 %	4.11	6.88
4	2.98	0.24 %	1.18 %	1.49 %	0.77 %	1.43 %	4.29	7.97
4	2.15	1.38 %	6.88 %	7.37 %	2.57 %	10.94 %	4	7.68
4	1.52	3.06 %	16.40 %	18.64 %	4.94 %	14.02 %	3.75	12.15
4	1.27	6.58 %	29.93 %	30.40 %	10.04 %	6.31 %	2.7	6.65
8	9.14	0.00 %	4.88 %	6.31 %	6.31 %	3.96 %	6.95	8.60
8	4.29	0.04 %	2.77 %	3.98 %	3.96 %	3.65 %	7.2	8.40
8	3.17	1.05 %	4.62 %	5.92 %	3.65 %	5.78 %	7.05	10.77
8	2.04	4.39 %	12.15 %	14.69 %	5.78 %	1.42 %	6.36	14.05
8	1.93	3.81 %	40.57 %	42.72 %	11.42 %	11.42 %	4.09	12.42
8	1.18	13.49 %	54.69 %	55.69 %	17.38 %	17.38 %	2.6	12.20

Table 2.6: Comparison between a hierarchical and a non-hierarchical clustering, using the look-ahead algorithm, searching one step ahead and computing the information theoretical-based score S_1.

82

Number of original groups	Noise B/W	% elements closer to other centres than their own	Clustering 1 % of misclassified elements		Clustering 2 % of misclassified elements		Average number of superclusters	Average number of branches in the tree
			Hierarchical tree	Grouping	k-means (k=10)	Grouping		
4	9.36	0.00 %	0.24 %	0.25 %	0.21 %	0.25 %	4.43	8.88
4	4.21	0.00 %	0.21 %	0.22 %	0.19 %	0.19 %	4.38	8.55
4	3.04	0.15 %	0.77 %	0.95 %	0.69 %	0.85 %	4.34	8.46
4	2.23	1.64 %	3.46 %	5.22 %	2.28 %	4.78 %	4.31	7.78
4	1.68	3.21 %	9.74 %	13.73 %	5.65 %	13.47 %	3.67	6.59
4	1.15	6.79 %	17.19 %	20.71 %	10.76 %	20.17 %	3.24	5.41
8	9.25	0.00 %	7.15 %	7.91 %	5.77 %	7.91 %	6.75	9.12
8	4.76	0.02 %	4.31 %	4.84 %	4.01 %	4.82 %	7.10	9.21
8	3.50	1.15 %	4.14 %	5.71 %	3.33 %	5.29 %	7.14	9.06
8	2.16	4.27 %	12.14 %	13.73 %	5.54 %	12.41 %	6.42	7.71
8	1.82	3.96 %	29.31 %	31.38 %	11.77 %	29.75 %	4.89	6.14
8	1.07	12.28 %	42.51 %	46.04 %	19.14 %	44.98 %	3.56	6.08

Table 2.7: Comparison between a hierarchical and a non-hierarchical clustering, with no look-ahead, computing the aesthetics-based score S_2.

83

Number of original groups	Noise B/W	% elements closer to other centres than their own	Clustering 1 % of misclassified elements		Clustering 2 % of misclassified elements		Average number of superclusters	Average number of branches in the tree
			Hierarchical tree	Grouping	k-means (k=10)	Grouping		
4	9.84	0.00 %	0.15 %	0.25 %	0.25 %	0.25 %	4.86	10.79
4	4.19	0.00 %	0.32 %	0.64 %	0.31 %	0.64 %	4.88	11.54
4	3.17	0.04 %	0.70 %	2.14 %	0.68 %	2.08 %	4.87	11.56
4	2.23	1.74 %	2.87 %	6.31 %	2.46 %	6.01 %	4.61	11.20
4	1.81	5.73 %	7.92 %	13.87 %	6.26 %	13.34 %	4.06	9.21
4	1.19	6.46 %	12.05 %	18.91 %	10.55 %	18.51 %	3.62	8.39
8	9.55	0.00 %	3.04 %	5.11 %	5.11 %	5.11 %	7.13	11.10
8	4.39	0.00 %	3.00 %	4.40 %	4.37 %	4.38 %	7.45	10.67
8	3.28	0.65 %	3.78 %	5.82 %	3.40 %	5.45 %	7.24	10.68
8	2.19	6.83 %	6.98 %	9.49 %	5.84 %	8.28 %	6.83	9.66
8	1.88	3.95 %	19.11 %	23.50 %	11.34 %	21.66 %	5.80	9.06
8	1.25	13.25 %	32.48 %	39.86 %	19.74 %	38.08 %	4.32	8.73

Table 2.8: Comparison between a hierarchical and a non-hierarchical clustering, using the look-ahead algorithm, searching one step ahead and computing the aesthetics-based score S_2.

84

h	Clustering 1 % misclassified elements		Clustering 2 % misclassified elements		Average number of superclusters	Average number of branches
	Hierarchical tree	Grouping	k-means (k=5)	Grouping		
0	5.03 %	5.13 %	0.74 %	1.98 %	3.02	3.43
0	23.29 %	23.62 %	13.72 %	20.50 %	2.49	2.85
0	39.66 %	40.32 %	25.30 %	31.67 %	1.79	2.03
0	45.43 %	45.88 %	31.78 %	37.87 %	1.55	1.85
0	47.89 %	48.07 %	36.85 %	40.75 %	1.36	1.48
0	48.44 %	48.62 %	39.40 %	42.59 %	1.39	1.59
1	0.61 %	1.00 %	0.74 %	0.98 %	3.81	6.39
1	15.53 %	18.25 %	13.18 %	18.16 %	3.76	8.76
1	29.30 %	32.20 %	25.48 %	28.17 %	3.56	8.98
1	33.99 %	37.18 %	31.87 %	34.25 %	3.57	9.42
1	37.78 %	41.23 %	35.96 %	37.82 %	3.45	8.73
1	40.90 %	44.29 %	39.62 %	41.44 %	3.18	8.97

Table 2.9: Comparison between a hierarchical and a non-hierarchical (k=5) clustering, using the look-ahead algorithm, with h= 0 and 1, and the information theoretical-based score S_1.

h	Clustering 1 % misclassified elements		Clustering 2 % misclassified elements		Average number of superclusters	Average number of branches
	Hierarchical tree	Grouping	k-means (k=5)	Grouping		
0	1.07 %	1.21 %	1.06 %	1.14 %	3.69	4.57
0	16.61 %	17.21 %	13.64 %	17.01 %	3.63	4.52
0	28.29 %	29.30 %	24.97 %	29.02 %	3.38	4.49
0	33.58 %	35.31 %	32.28 %	35.15 %	3.68	5.28
0	37.70 %	39.40 %	35.55 %	39.22 %	3.47	5.11
0	41.54 %	42.83 %	40.59 %	42.97 %	3.52	5.16
1	1.54 %	1.80 %	1.68 %	1.68 %	4.27	6.52
1	13.41 %	15.13 %	13.82 %	15.32 %	4.12	6.68
1	25.88 %	28.45 %	25.19 %	28.46 %	3.98	6.85
1	32.28 %	34.76 %	32.05 %	34.61 %	3.90	6.82
1	36.32 %	38.20 %	36.01 %	38.09 %	3.91	6.89
1	39.19 %	41.66 %	39.15 %	41.34 %	3.85	7.14

Table 2.10: Comparison between a hierarchical and a non-hierarchical (k=5) clustering, using the look-ahead algorithm, with h= 0 and 1, and the aesthetics-based score S_2.

not look ahead any further, due to the fact that S_2 is less restrictive than S_1 (see the summary in table 2.10). The average number of superclusters is close to 3, and the number of branches needed to identify them is between 4 and 5. The percentages of misclassification are very similar to those obtained for S_1 and h = 1. On the other hand, when h=1, the number of superclusters is around 4 and the percentage of misclassification does not decrease significantly.

In brief, it appears that the visualization aesthetics-based scoring function outperforms the MDL principle for a small number of look-ahead steps. Therefore it can be used to improve the execution time (in our implementation it takes seconds to compare clusters of thousands of genes without look-ahead, and minutes with look-ahead).

2.4.2. Results in real data

We also tested the algorithm on real data. As an example, Figure 2.17 shows the comparison of a hierarchical clustering (where we selected correlation-based distance and the average linkage method) to a *k*-means clustering (with correlation-based distance and k=5 clusters) of the *Schizosaccharomyces pombe* stress response dataset (Chen *et al.*, 2003), (ArrayExpress accession number E-MEXP-29).

This set, available at http://www.sanger.ac.uk/PostGenomics/S_pombe/projects/stress/, characterizes the changes in expression profiles of all known and predicted genes of the fission yeast S. pombe in response to five stress conditions (oxidative stress caused by hydrogen peroxide, heavy metal stress caused by cadmium, heat shock caused by temperature increase to 39^oC, osmotic stress caused by sorbitol, and DNA damage caused by the alkylating agent methylmethane sulfonate), measured using DNA microarrays in three strains (wild-type and two mutants: $sty1\Delta$ and $atf1\Delta$). The data set was filtered so that we retain the 140 most varying ones ($\geq$ 0.9 standard deviations in 60 % of the hybridizations). Then we applied the algorithm; we used no look-ahead, and scoring function S_2. The highly-responding (yellow) cluster is highly enriched with the so-called CESR (Core Environmental Stress Response) genes.

These are genes that are up-regulated in at least four of the five conditions or down-regulated in at least three of the five conditions. The up-regulated cluster contains 12 of such genes out of 64. Near the middle of the hierarchical tree, the cluster of genes of high negative response (blue) are mostly mitochondrial ones. The down-regulated CESR genes are also enriched (25/62).

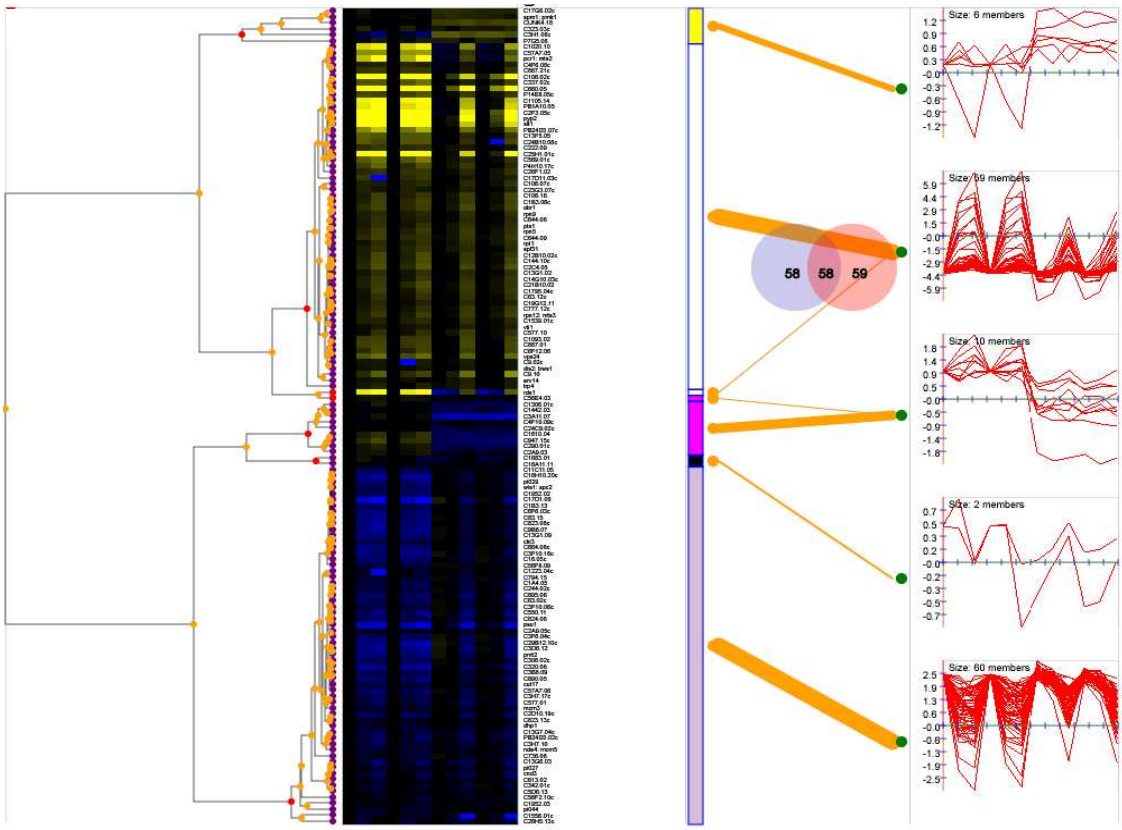


Figure 2.17: A comparison of the hierarchical clustering of (correlation-based distance, average linkage) ~ 140 most varying genes (≥ 0.9 standard deviations in 60 % of the hybridizations) in the *S. pombe* stress response dataset to a k-means clustering of the same dataset (correlation-based distance, k=5).

This example shows that, when applying the comparison algorithm to real gene expression data, we can obtain biologically meaningful results, which helps expert biologists to explore their data.

2.5. Implementation in Expression Profiler

The algorithms described in previous sections are available through the online gene expression data analysis tool Expression Profiler (EP) (Kapushesky *et al.*, 2004) at the European Bioinformatics Institute (EBI) (http://www.ebi.ac.uk/expressionprofiler). A user can upload a gene expression matrix together with associated annotations to the EP platform, where a number of analytical components are available, including data transformation, sub-selection, statistical tests and clustering, both hierarchical and k-means. The Clustering Comparison component, under the "Clustering" menu allows the user to run the algorithm and view the visualization of the output. The user is presented with the analysis history of the current dataset and can select a pair of previously performed clusterings to be compared. Alternatively, the user can select a dataset to be analysed and set up two sets of clustering parameters. In this case the two clusterings will be executed and then their results will be compared and the visual display will be produced. For running hierarchical clusterings the user needs to specify a distance measure (e.g., Euclidean or Pearson correlation, etc.) and type of linkage (complete, average, etc.). Similarly, the distance measure should be specified for flat clusterings, as well as the number k for k-means.

Adjustable clustering comparison algorithm parameters are also presented. The user can specify the number of steps to use in the look-ahead tree search and the type of scoring function to use (information-theoretic or aesthetics-based).

The underlying clustering comparison algorithm is implemented in R. The source code is available as part of the open-source distribution of Expression Profiler. The EP data analysis tool also provides a visualization of the comparison (see Figure 2.17), displaying side-by-side the tree, with optimal cut-off points marked in red, the expression heatmap, gene annotations, the set of k clusters resulting from the flat clustering, and the clustering comparison correspondence. The latter is depicted as lines of varying thickness, mapping sub-branches of the tree to flat clustering superclusters. Line thickness is proportional to the number of elements common to

both sets; by placing the mouse cursor over a line, a Venn diagram is displayed with the exact numbers – how many elements are in the cluster generated by cutting the respective branch, how many in the corresponding flat cluster and how many of these two overlap.

In summary, from the results shown in this chapter we can conclude that the novel method that we have developed allows the user to explore relationships between different clustering results, including hierarchical clusterings without any *a priori* given cut-offs. Although the biological relevance of the comparison results depends on the clusters that are compared, in some cases it is possible to improve the original clustering.

Capítulo 3

Refinamientos robustos de k-medias mediante profundidad

Resumen

Muchas técnicas iterativas son sensibles a las condiciones iniciales, por lo que pueden quedar atrapadas en óptimos locales. Este capítulo explora dos métodos simples y computacionalmente rápidos que permiten el refinamiento de los puntos iniciales de k-medias para agrupar las observaciones de un conjunto dado. Se basan, respectivamente, en alternar k-medias con la búsqueda del dato más profundo (más representativo) de cada cluster, y en combinar bootstrap con la búsqueda de los datos más profundos de cada cluster en el espacio de los centros bootstrap. Además, combinada con la construcción de regiones de Voronoi generalizadas a lugares geométricos convexos, esta segunda alternativa puede proporcionar un clustering "no rígido" que asigna a cada elemento del conjunto de datos un índice de pertenecia a cada cluster. Esto es una alternativa útil en ciertos campos, como en expresión génica, donde los datos analizados pueden pertenecer simultáneamente a varios clusters. Estos métodos se han probado en observaciones simuladas, incluyendo modelos contaminados, y en datos reales, y han demostrado ser eficientes y más rápidos que el método DDclust, propuesto en Jörnsten (2004).

3.1. Introducción

Es bien sabido que k-medias y otras técnicas iterativas (e.g. el algoritmo EM) son especialmente sensibles a las condiciones iniciales, i.e., a menudo el proceso puede obtener sólo uno de los óptimos locales del problema considerado, lo cual puede conducir a conclusiones muy distintas en el análisis de un mismo conjunto de observaciones. Puesto que se ha demostrado que el problema de obtener un estado inicial globalmente óptimo es de complejidad NP (Garey y Johnson, 1979; Garey *et al.*, 1982), resulta más práctico y valioso estudiar métodos de inicialización dirigidos a encontrar una solución subóptima del problema de clustering.

Existen varios métodos de inicialización de algoritmos iterativos, que pueden ser divididos principalmente en tres grandes categorías (He *et al.*, 2004):

- métodos de muestreo aleatorio, que son, probablemente, los más usados en la literatura y siguen una forma "ingenua" de determinar los clusters iniciales, o bien seleccionando aleatoriamente observaciones muestrales, o bien eligiendo parámetros aleatorios generados de forma no heurística a partir de los datos. Como ejemplo, véanse Forgy (1965) o MacQueen (1967).

- métodos de optimización de distancias, cuyo objetivo es optimizar las distancias entre los clusters iniciales para obtener en el resultado una separación inter-cluster satisfactoria Véanse como ejemplo (Tou y González, 1974; Katsavounidis *et al.*, 1994).

- métodos de estimación de la densidad, que se basan en la suposición de que los datos siguen una distribución de mixturas gaussianas e intentan identificar en ellos áreas densas que ayuden a crear clusters compactos (Kaufman y Rousseeuw, 1990; Al-Daoud y Roberts, 1994).

Existe además otra categoría de estudios en los cuales estamos interesados y que tratan el *refinamiento del estado inicial* de los métodos de clustering. Estos estudios incluyen un refinamiento propuesto por Bradley y Fayyad (1998) que elige inicialmente J submuestras aleatorias de pequeño tamaño de los datos, $S_i, i = 1, ..., J$. Entonces, las submuestras se agrupan mediante k-medias, imponiendo la condición de que en los clusters que terminen vacíos se reasignarán sus correspondientes centros iniciales, y la submuestra será reagrupada. Los conjuntos $CM_i, i = 1, ..., J$, son los clusterings que resultan de las submuestras, y conforman el conjunto CM. Finalmente, CM es agrupado para cada i, usando k-medias inicializada con CM_i, lo cual proporciona la solución FM_i. Los puntos iniciales (semillas) refinados se escogen como la solución FM_i que tiene menor distorsión sobre el conjunto CM.

También existen algoritmos genéticos diseñados para refinar los puntos iniciales de k-medias, pero tienen una gran carga computacional; por ejemplo, véase el método descrito por Laszlo y Mukherjee (2006) para datos de baja dimensión.

En una línea de investigación diferente, Jörnsten (2004) propuso un método de clustering robusto, el algoritmo DDclust , basado en el concepto de profundidad, para resolver el problema de minimizar la suma de las distancias L_1 de las observaciones al representante más cercano de los clusters.

Una profundidad es una medida de lo central o exterior que es una observación dada con respecto a una nube de puntos o a una distribución. Las funciones de profundidad introducen un orden desde el centro hacia fuera en los datos multidimensionales. En Estadística, la motivación y la necesidad de generalizar la mediana y el rango es muy natural, puesto que la media no es una medida robusta de posición central. La profundidad considerada en Jörnsten (2004) es la profundidad L_1 de Vardi y Zhang (2000), cuya interpretación estadística es la probabilidad que se necesita en un punto z para que dicho punto sea la mediana L_1 multivariante del

cluster – i.e., un representante robusto del cluster. Esta es la primera propuesta de un método de clustering en el que se usa el concepto de profundidad, y combina la búsqueda de estas medianas multivariantes con la asignación a los vecinos más próximos, además de incluir una técnica de "recocido simulado" (simulated annealing), para evitar quedarse atrapado en un mínimo local.

En este capítulo nos proponemos obtener un procedimiento de refinamiento del estado inicial de k-medias, combinando ideas de Bradley y Fayyad (1998) y Jörnsten (2004), utilizando una generalización finito-dimensional de la mediana univariante diferente de la empleada en Jörnsten (2004). La profundidad que utilizamos es la denominada profundidad por bandas generalizada, descrita en López-Pintado y Romo (2006), aunque cualquier definición de profundidad de datos multidimensionales podría utilizarse en nuestro método.

Una primera propuesta para refinar k-medias es la siguiente. Supongamos que hemos agrupado el conjunto X en k grupos mediante k-medias, con los centros iniciales elegidos aleatoriamente. Sea cual sea la profundidad elegida, hallamos el dato más profundo (más central) $\mathcal{D}_i$ en cada uno de los k clusters obtenidos, y usamos estos nuevos puntos como centros iniciales en el algoritmo de k-medias. Si esto conlleva una reducción en la función objetivo, continuamos iterando con los nuevos puntos más profundos que obtengamos usados como centros iniciales; en otro caso, mantenemos el resultado de k-medias que proporciona el valor mínimo en la función objetivo. De esta forma, no podemos obtener resultados peores que los derivados de k-medias en términos de la función objetivo. Sin embargo, hay que observar que los conglomerados reales con que nos encontramos en la práctica no siempre minimizan la función objetivo y, así, valores más pequeños de la distorsión se corresponden con un número superior de observaciones mal asignadas, y viceversa. Los resultados obtenidos en este enfoque muestran que este procedimiento de refinamiento funciona bien para una gran variedad de conjuntos de datos, pero puede

ocurrir que k-medias encuentre un (mal) óptimo local del cual nuestro método no es capaz de "escapar". Por ello, proponemos una alternativa basada en extraer muestras bootstrap del conjunto de observaciones para obtener una colección de centros C, que es agrupada mediante k-medias; finalmente, en cada cluster (de centros) se halla el dato más profundo, para usarlos como puntos iniciales de k-medias. Este procedimiento proporciona un método muy eficiente y robusto, que supera a k-medias en todos los casos y se comporta de manera semejante a la de DDclust, pero es mucho más rápido.

Este capítulo está organizado como sigue. La sección 3.2 proporciona una manera eficiente de calcular la profundidad por bandas generalizada, y describe dos métodos nuevos, GBDRk (k-medias refinada por la profundidad por bandas generalizada) en la subsección 3.2.2, y BRk (k-medias refinada por bootstrap) en la subsección 3.2.3, que permiten el refinamiento de los puntos iniciales de k-medias. Estos dos algoritmos se basan, respectivamente, en alternar k-medias con la búsqueda del dato más profundo en cada cluster y en combinar resultados de bootstrap sobre k-medias con la búsqueda de los puntos más profundos de cada cluster en el espacio de los centros de los clusters. El comportamiento de ambos métodos se compara con el de otros métodos utilizados en la literatura, usando tanto datos simulados como reales. Finalmente, en la sección 3.4 describimos cómo emplear el método BRk para proporcionar un clustering no rígido de las observaciones, lo que es útil en campos como el análisis de datos de expresión génica, en el que los genes pueden pertenecer a más de un cluster.

3.2. Métodos

3.2.1. Un cálculo eficiente de la profundidad por bandas generalizada

En Estadística no paramétrica existen diversas medidas de profundidad como generalizaciones del rango con el objetivo de complementar el análisis multivariante clásico, primero por Tukey (1975), seguido posteriormente por Oja (1983), Liu (1990), y Donoho y Gasko (1992), entre otros. Sin embargo, una de las mayores desventajas en todas las definiciones de profundidad de datos es el coste computacional asociado, lo que restringe su uso a conjuntos de observaciones de baja dimensión. Nuestro procedimiento, que será descrito en la sección siguiente, permite el uso de cualquier definición de profundidad, aunque nosotros hemos empleado la profundidad por bandas generalizadas descrita en López-Pintado y Romo (2006), que resulta apropiada para datos de alta dimensión, para reducir el tiempo de computación. Sea $Y = \{y_1, \ldots, y_n\}$ un conjunto de n puntos d-dimensionales, y denotemos por $y_{i,k}, i = 1, ..., n$, la k-ésima componente del elemento y_i. La versión finito-dimensional de la profundidad por bandas generalizada GBD de un punto y se define como la proporción media de las componentes de y que se encuentran entre las correspondientes componentes de todos los posibles pares de elementos de Y. Más formalmente:

$$GBD(y) = \frac{1}{\binom{n}{2}} \sum_{1 \le i_1 < i_2 \le n} \frac{1}{d} \sum_{k=1}^{d} I_{\{min(y_{i_1,k}, y_{i_2,k}) \le y_k \le max(y_{i_1,k}, y_{i_2,k})\}} \tag{3.2.1}$$

Obsérvese que la profundidad por bandas generalizada se puede calcular de forma muy eficiente rescribiendo la expresión 3.2.1 como se describe en la siguiente proposición.

Proposición 3.1 *Dados un conjunto de observaciones* $Y = \{y_1, ..., y_n\}$, *y un punto* $y_i \in Y$ *con-*

sideremos las matrices $\mathbf{Y} = \begin{pmatrix} y_{1,1} & \cdots & y_{1,d} \\ \vdots & & \vdots \\ y_{n,1} & \cdots & y_{n,d} \end{pmatrix}$ *and* $\widetilde{\mathbf{Y}} = \begin{pmatrix} y_{(1),1} & \cdots & y_{(1),d} \\ \vdots & & \vdots \\ y_{(n),1} & \cdots & y_{(n),d} \end{pmatrix}$, *donde cada columna en* $\mathbf{Y}$ *ha sido organizada en orden creciente. Sea* l_k *el menor índice en la* k*-ésima columna de* $\widetilde{\mathbf{Y}}$ *que verifica* $y_{i,k} = y_{(l_k),k}$, *y sea* η_k *la multiplicidad del valor* y_k *en la misma. Entonces, la profundidad por bandas generalizada del punto* y_i *está dada por*

$$\frac{1}{d \times \binom{n}{2}} \sum_{k=1}^{d} \left((n - l_k + 1)(l_k - 1 + \eta_k) - \eta_k^2 + \binom{\eta_k}{2} \right). \tag{3.2.2}$$

Demostración: Nótese que

$$\begin{aligned}
GBD(y_i) &= \frac{1}{\binom{n}{2}} \sum_{1 \le i_1 < i_2 \le n} \frac{1}{d} \sum_{k=1}^{d} I_{\{min(y_{i_1,k}, y_{i_2,k}) \le y_{i,k} \le max(y_{i_1,k}, y_{i_2,k})\}} = \\
&= \frac{1}{\binom{n}{2}} \frac{1}{d} \sum_{k=1}^{d} \sum_{1 \le i_1 < i_2 \le n} I_{\{min(y_{i_1,k}, y_{i_2,k}) \le y_{i,k}\}} I_{\{max(y_{i_1,k}, y_{i_2,k}) \ge y_{i,k}\}} = \\
&= \frac{1}{\binom{n}{2}} \frac{1}{d} \sum_{k=1}^{d} \sum_{1 \le i_1 < i_2 \le n} \left(I_{\{min(y_{i_1,k}, y_{i_2,k}) < y_{i,k}\}} + I_{\{min(y_{i_1,k}, y_{i_2,k}) = y_{i,k}\}} \right) \times \\
&\times \left(I_{\{max(y_{i_1,k}, y_{i_2,k}) > y_{i,k}\}} + I_{\{max(y_{i_1,k}, y_{i_2,k}) = y_{i,k}\}} \right) = \\
&= \frac{1}{d \times \binom{n}{2}} \sum_{k=1}^{d} [(l_k - 1) \cdot (n - (l_k + \eta_k - 1)) + (l_k - 1) \cdot \eta_k + \\
&+ \eta_k \cdot (n - (l_k + \eta_k - 1)) + \binom{\eta_k}{2}] = \\
&= \frac{1}{d \times \binom{n}{2}} \sum_{k=1}^{d} \left((n - l_k + 1)(l_k - 1 + \eta_k) - \eta_k^2 + \binom{\eta_k}{2} \right).
\end{aligned}$$

■

Este cálculo reduce significativamente el coste computacional. Por ejemplo, implementar la

profundidad en R implica una reducción con respecto al cálculo de la correspondiente proporción para todos los pares, que es $O(d.n^2)$, a $O(n^{4/3})$ para reordenar la matriz $\mathbf{Y}$ usando una variante de (Sedgewick, 1986) más $O(d)$ para calcular el valor 3.2.2.

3.2.2. Alternancia de k-medias con la búsqueda de datos más profundos

Consideremos un conjunto X en $\mathcal{R}^d$, cuyos elementos se quieren agrupar en k grupos. Denotemos por $\mathcal{C}_1, \ldots \mathcal{C}_k$ los centros de estos grupos y por Δ la función objetivo o distorsión, dada por la suma de las distancias al cuadrado entre cada punto $x \in X$ y el centro más cercano $\mathcal{C}_{i(x)}$. El método de k-medias refinado por la profundidad por bandas generalizada (GBDRk) se describe formalmente más abajo mediante un pseudo-código. Los valores de entrada para el método son el conjunto de observaciones X, el número de conglomerados k que se desea obtener y el número máximo de iteraciones, $max.iter$.

Cuadro 3.2.2.1. Algoritmo iterativo de selección de centros iniciales basado en el concepto de profundidad de datos.

no.iterations $\leftarrow 0$

WHILE ($no.iterations < max.iter$) DO:

1. no.iterations $\leftarrow$ no.iterations + 1
2. Usar k-medias en X iniciado con k puntos elegidos aleatoriamente para obtener k clusters, $I_1^1, ..., I_k^1$, con centros respectivos $\mathcal{C}_1^1, \ldots \mathcal{C}_k^1$.
3. Calcular la suma de distancias al cuadrado de cada punto a su centro más próximo Δ_1.

4. Encontrar el punto más profundo de cada cluster: $\mathcal{D}_1^1 \in I_1^1, \ldots, \mathcal{D}_k^1 \in I_k^1$.

5. Usar k-medias, con puntos iniciales $\mathcal{D}_1^1, \ldots \mathcal{D}_k^1$, para obtener k nuevos clusters $I_1^2, ..., I_k^2$ con centros $\mathcal{C}_1^2, \ldots \mathcal{C}_k^2$.

6. Calcular la suma de las distancias al cuadrado Δ_2 para el nuevo clustering.

7. IF $\Delta_2 < \Delta_1$

   THEN

   $I_j^1 \leftarrow I_j^2$, $j = 1, ..., k$; $\Delta_1 \leftarrow \Delta_2$

   ir al paso 3

   OTHERWISE

   dar como resultado el clustering $I_1^1, ..., I_k^1$ y parar

Este algoritmo de refinamiento da como resultado un clustering que proporciona un valor de la función objetivo que es al menos tan buena como la que da k-medias, porque si la utilización de los datos más profundos como centros iniciales no mejora esta función, entonces conservamos el resultado original de k-medias.

En general, los resultados obtenidos en datos simulados y reales con este procedimiento demuestran que se comporta mejor que k-medias, pero para algunos conjuntos de datos, el método propuesto no es capaz de mejorar algunos malos resultados, en los que k-medias se queda "atascada". Una alternativa podría ser utilizar una técnica de recocido simulado (simulated annealing) para escapar de los óptimos locales, como en el caso de DDclust; esto será objeto de investigación futura. En su lugar, en la sección siguiente describiremos un en-

foque diferente basado en hacer bootstrap sobre el conjunto de observaciones X, combinado con la búsqueda del dato más profundo de cada cluster, que resulta ser muy robusto y mejora a k-medias, a la vez que es comparable con DDclust, con la ventaja de ser mucho más rápido.

3.2.3. Combinación de bootstrap con la búsqueda de datos más profundos

Como se mencionó en la sección 3.2.2, estamos interesados en encontrar un método alternativo que refine las semillas a un nuevo conjunto de puntos que estén próximos a los máximos (modas) de la función de densidad de los datos. De forma similar a Bradley y Fayyad (1998), proponemos una heurística basada en remuestrear (en particular, usando bootstrap) los datos B veces, y utilizar k-medias estándar con k clusters en cada muestra bootstrap, $b = 1, \ldots, B$, para obtener un conjunto $\mathcal{C}$ de $b \cdot k$ centros estimados; finalmente, buscamos k nuevos clusters en $\mathcal{C}$, y usamos el dato más profundo de cada cluster como semillas iniciales.

La heurística se basa en el concepto de bagging (Breiman, 1996) usado en aprendizaje supervisado, que combina las clasificaciones predichas a partir de diferentes datos de entrenamiento mediante algún esquema de votación simple; la clasificación final obtenida para cada dato es la correspondiente a la más frecuente entre las predichas.

Existe trabajo previo de bagging aplicado a clustering. Leisch (1999) diseñó un "bagged clustering" que combina métodos de partición y métodos jerárquicos. Su algoritmo obtiene una colección de conjuntos de entrenamiento usando bootstrap, a continuación utiliza cualquier clustering no jerárquico en cada uno de estos conjuntos, y finalmente combina las particiones resultantes usando clustering jerárquico. Dudoit y Fridlyand (2003) propusieron otros dos métodos que extienden el concepto de bagging al clustering. Uno de éstos se basa en un criterio de votación y el otro en la creación de una nueva matriz de disimilaridades. En nuestro procedi-

miento de aprendizaje no supervisado, el papel de la clasificación más votada es desempeñado por los clusters existentes en $\mathcal{C}$, y usamos como representantes iniciales de los clusters en el espacio original el dato más profundo de tales clusters.

Como en Leisch (1999), nuestro método, en primer lugar, transforma el problema de clustering en el espacio de los datos originales en un nuevo problema en el espacio de los centros producidos por k-medias. Supongamos que el conjunto de observaciones X se genera por una distribución F desconocida, con densidad f. Asumamos que f es multimodal y que cada moda corresponde a un cluster. Si los centros producidos por k-medias se concentran en las modas de f, entonces los nuevos clusters son más pequeños en el sentido euclidiano –y de hecho, para cualquier distancia– que los originales, con mayor probabilidad. Esto puede verse en el Teorema 2 de Leisch (1999). La diferencia con el enfoque de Leisch es que nosotros fijamos a priori el número de clusters que estamos buscando (recordemos que nuestro objetivo es refinar los puntos iniciales de k-medias) y que la búsqueda del dato más profundo de cada cluster en un contexto multivariante es más robusto, lo cual atenúa la influencia de muestras bootstrap poco representativas.

Nuestro algoritmo está descrito mediante pseudo-código en el cuadro 3.2.3.1. De nuevo, denotamos por X un conjunto de observaciones en $\mathcal{R}^d$, cuyos elementos van a ser agrupados en k clusters. Los datos de entrada son el conjunto X, el número de clusters k que vamos a obtener y el número de replicaciones bootstrap B.

1. FOR (b in $1 : B$) DO:

   - Obtener la muestra bootstrap X^b del conjunto X.

   - Usar k-medias con k puntos iniciales, elegidos aleatoriamente, sobre el conjunto X^b para obtener k clusters, $I_1^b, ..., I_k^b$, con centros respectivos $\mathcal{C}_1^b, \ldots \mathcal{C}_k^b$. Si alguno de los clusters queda vacío, volver a usar k-medias con diferentes puntos iniciales.

2. Usar k-medias en $\mathcal{C} = \{\mathcal{C}_i^b, i = 1, ..., k, b = 1, ..., B\}$ para obtener k clusters, $I_1^*, ..., I_k^*$.

3. Encontrar el punto más profundo de cada cluster: $\mathcal{D}_1 \in I_1^*, \ldots, \mathcal{D}_k \in I_k^*$.

4. Usar k-medias con puntos iniciales $\mathcal{D}_1^1, \ldots \mathcal{D}_k^1$.

5. Dar como resultado el clustering $I_1, ..., I_k$.

Box 3.2.3.1. Algoritmo de selección de los centros iniciales basado en la combinación de bootstrap con el concepto de profundidad de datos.

Observemos que una pequeña variación en el método permite transformar el procedimiento de refinamiento en un algoritmo de clustering en sí mismo: en lugar de usar el dato más profundo de cada cluster en el espacio de los centros $\mathcal{D}_1 \in I_1^*, \ldots, \mathcal{D}_k \in I_k^*$ como puntos iniciales de k-medias, éstos pueden ser utilizados como centroides (finales) y podemos asignar cada elemento de X al centroide más próximo, definiendo así el clustering. Esto es equivalente a usar sólo la primera iteración de k-medias en el Paso 4 del método BRk, y en general se deberían obtener resultados muy similares.

Se puede demostrar que el método BRk, bajo ciertas condiciones de simetría y de regularidad, es consistente, simplemente combinando los resultados correspondientes de k-medias

(Pollard, 1982), de bootstrap de M-estimadores (Arcones y Giné, 1992) y de datos más profundos (López-Pintado y Romo, 2006).

Teorema 3.1 *Sea P una distribución simétrica en $\mathbb{R}^d$ y supongamos*

i) el vector $\mu = (\mu_1, \ldots, \mu_k)$ que minimiza la suma de cuadrados intra-clusters poblacional $W(\cdot)$ es único excepto por permutación de sus coordenadas;

ii) $P\|x\|^2 < \infty$;

iii) P tiene una densidad continua f.

Entonces, el vector de los puntos más profundos $\boldsymbol{D}$ calculados en el método BRk es consistente para estimar el vector μ.

Demostración: En primer lugar, siguiendo los resultados de Arcones y Giné (1992) para la aproximación bootstrap de M-estimadores, los centros boostrap $\{\mathcal{C}_1^b, \ldots \mathcal{C}_k^b\}$, $b = 1, \ldots, B$ calculados en el Paso 1 son estimadores consistentes de las localizaciones de los centros, i.e., excepto por permutación de coordenadas se tiene:

$$\mathcal{C}_i^b \to \mu_i, \quad b = 1, \ldots, B,\ i = 1, \ldots, k.$$

A continuación, observemos que en el espacio de los centros (bootstrap), las condiciones i) y ii) se verifican, por lo que, usando el teorema central del límite de Pollard (1982), aplicar k-medias en este espacio implica que las correspondientes localizaciones de los centros convergen en probabilidad al vector μ.

Finalmente, como la profundidad por bandas generalizada es consistente, los correspondientes datos más profundos $\mathcal{D}_1^1, \ldots \mathcal{D}_k^1$ convergen a los datos más profundos del espacio inicial, que

coinciden con μ si la distribución es simétrica. ■

En general, para datos generados, BRk tiene un comportamiento muy robusto, similar e incluso mejor en ocasiones que los de DDclust, siendo ambos los mejores de los métodos analizados. BRk consigue disminuir tanto la distorsión como el porcentaje de elementos mal asignados por k-medias, y proporciona sus resultados en mucho menos tiempo que DDclust.

3.3. Resultados

3.3.1. Resultados en datos simulados

Los experimentos de simulación fueron realizados como sigue. Para cada modelo particular creamos cincuenta conjuntos de datos y usamos el algoritmo de k-medias estándar, el algoritmo PAM, el algoritmo CLARA, DDclust, k-medias refinada por la profundidad por bandas generalizada (GBDRk) y k-medias refinada por bootstrap (BRk). En todos los algoritmos usamos el mismo número de clusters, que correspondía con el número real de grupos simulados. Para evaluar el comportamiento de estos métodos en términos de la distorsión, calculamos su valor obtenido para cada conjunto. Observemos, no obstante, que PAM, CLARA y DDclust están diseñados para minimizar la suma de distancias y no la distorsión. Para evaluar el comportamiento de estos métodos en términos del error de asignación, calculamos las tasas de error y las representamos en diagramas de caja y en tablas que contienen los cuartiles, así como los valores mínimo y máximo.

Para evaluar el comportamiento de los métodos en presencia de datos atípicos, también empleamos diferentes modelos contaminados. Un modelo contaminado se construye como sigue: la mayor parte de los puntos de un cluster se obtienen de una distribución "progenitora" $\mathcal{P}_p$, mientras que los restantes puntos, en una proporción de ϵ, proceden de otra distribución

denotada por $\mathcal{P}_c$. Empleamos los siguientes modelos:

a) Contaminación simétrica: $\mathcal{P}_c$ tiene la misma media pero diferente desviación estándar que $\mathcal{P}_p$.

b) Contaminación asimétrica: $\mathcal{P}_c$ tiene diferente media que $\mathcal{P}_p$. Aumentar la diferencia entre la media de $\mathcal{P}_p$ y la de $\mathcal{P}_c$ crea un mayor grado de contaminación asimétrica.

Además, se puede crear un mayor grado de contaminación por datos atípicos aumentando la proporción de muestreo de $\mathcal{P}_c$.

Ejemplo 3.1 *Modelo 1 de tamaño 50*

Generamos 50 conjuntos de datos con 6 conglomerados y un nivel de ruido $\sigma = 4$ (para niveles de ruido más bajos, el comportamiento de todos los métodos era muy similar, por lo que no mostramos los resultados). El mejor método es BRk, seguido por DDclust, los cuales obtienen bajas tasas de error. Sin embargo, DDclust une dos de los clusters en algunos casos (mostrados como datos atípicos en el diagrama de caja). PAM y CLARA son menos robustos, pero obtienen mejores resultados que GBDRk y k-medias, que, en cualquier caso, es el peor método. Los mínimos, los máximos y los cuartiles se muestran en la tabla 3.1. Con respecto a la distorsión, los mejores valores se obtienen con BRk, que reduce los de k-medias, PAM, CLARA y DDclust en 24, 47, 47 y 38 de los 50 conjuntos generados, respectivamente, seguido por GBDRk, que decrece dichos valores en 11, 29, 30 y 28 casos, respectivamente.

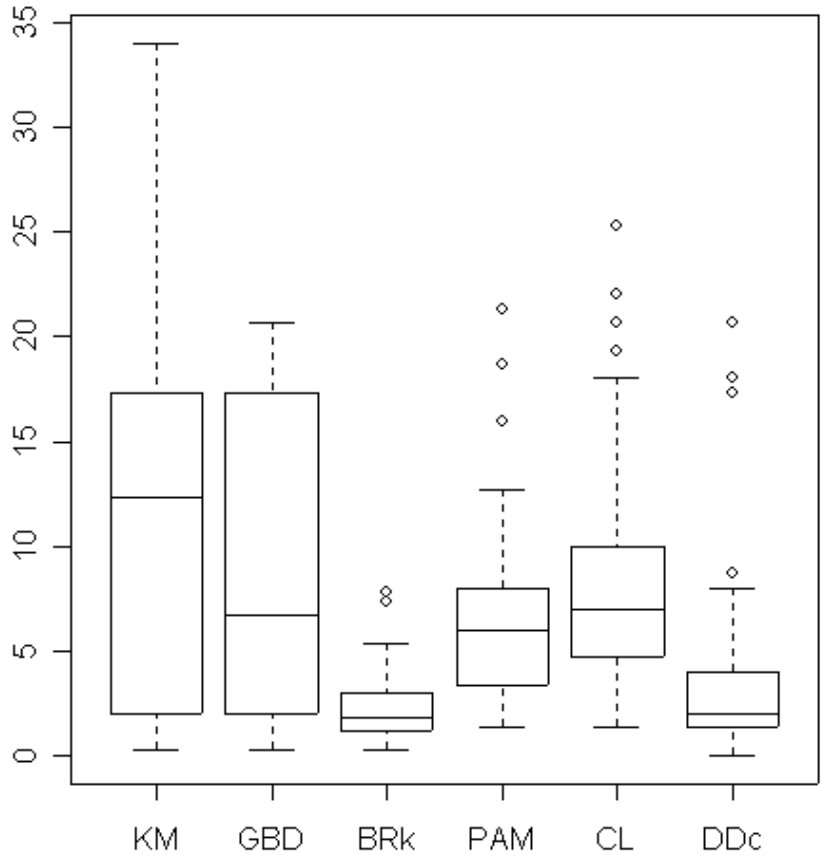


Figura 3.1: Diagramas de caja para las tasas de error obtenidas con los seis métodos para conjuntos de datos generados según el Modelo 1, con 6 grupos de tamaño 50 y un nivel de ruido de $\sigma = 4$.

Tabla 3.1: Valores mínimo, máximo y cuartiles de las tasas de error obtenidas para el Modelo 1, con 6 grupos y $\sigma = 4$.

Método	Mínimo	Primer cuartil	Mediana	Tercer cuartil	Máximo
k-medias	0.22	2.00	12.33	17.33	34.00
GBDRk	0.22	2.00	6.67	17.33	20.67
BRk	0.22	**1.11**	**1.78**	**3.00**	**7.78**
PAM	1.33	3.50	6.00	7.83	21.33
CLARA	1.33	4.83	7.00	10.00	25.33
DDclust	**0.00**	1.33	2.00	4.00	20.67

■

El ejemplo 3.1 ilustra aquellas situaciones en las que el uso de los datos más profundos en los clusters obtenidos con k-medias como puntos iniciales no puede evitar el problema de fusionar dos clusters que proceden de diferentes distribuciones, mientras que divide en dos un grupo. Esto se puede entender fácilmente observando la figura 3.2, en la que las cruces rojas muestran los centros finales de k-medias y los círculos azules muestran los correspondientes datos más profundos. Sin embargo, BRk demuestra ser muy robusto, consiguiendo tasas de error de asignación muy bajas. Otra alternativa podría ser usar una técnica de recocido simulado para escapar del óptimo local, como en el caso de DDclust.

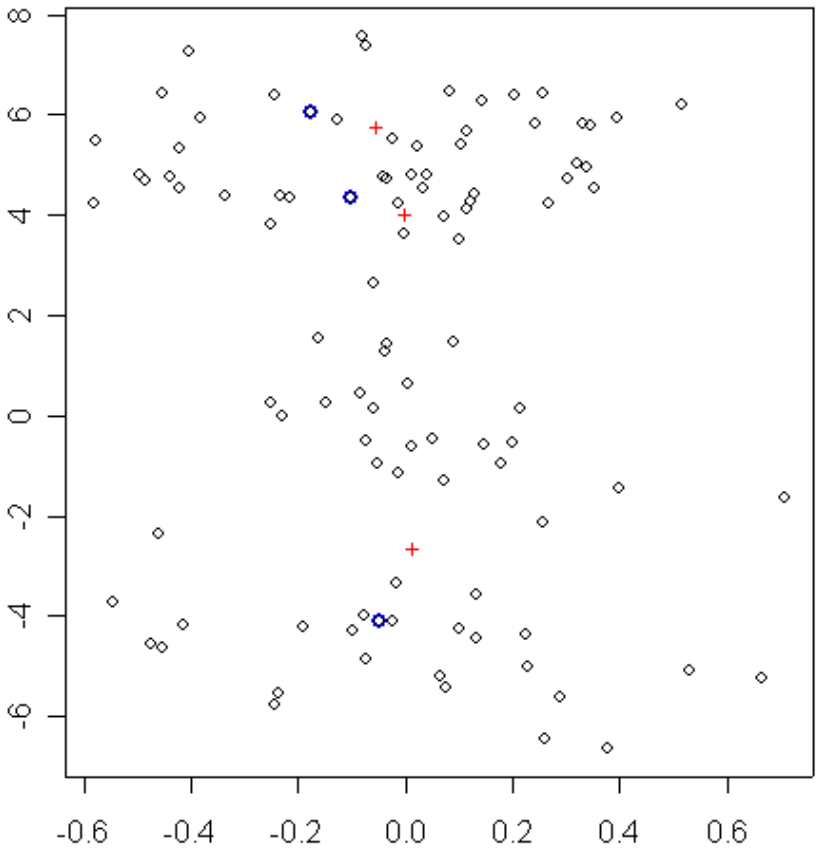


Figura 3.2: Conjunto de datos procedente de una mixtura de tres distribuciones gausianas bivariantes con medias linealmente dependientes localizadas en $\mu_1 = (0, 0), \mu_2 = (0, 5)$ y $\mu_3 = (0, -5)$. Los tres centros dados por k-medias se representan con cruces rojas, mientras que los datos más profundos en los clusters correspondientes están señalados con círculos azules. El uso de estas observaciones más profundas como semillas iniciales en k-medias produce los mismos centros y el método GBDRk no puede escapar del mínimo local.

Ejemplo 3.2 *Cuatro clusters de tamaño 50, en dimensión 3, localizados en los vértices (0,0,0), (3,0,0), (0,3,0) y (0,0,3) de un tetraedro, con cada componente generada mediante variables exponenciales de media 1.*

En este caso, todos los métodos se comportan de forma muy similar, siendo BRk el que proporciona los cuartiles más pequeños. Este método es seguido por k-medias y GBDRk. Sin embargo, en tres de los casos, k-medias une dos clusters, por lo que aparecen tasas de error elevadas. GBDRk consigue mejorar estos resultados en uno de los tres casos, pero falla en los otros dos. Sólo en ese caso la distorsión se reduce, mientras que el número de casos de los 50 en los que el correspondiente valor obtenido con PAM, CLARA y DDclust se mejora es 45, 48 y 43, respectivamente.

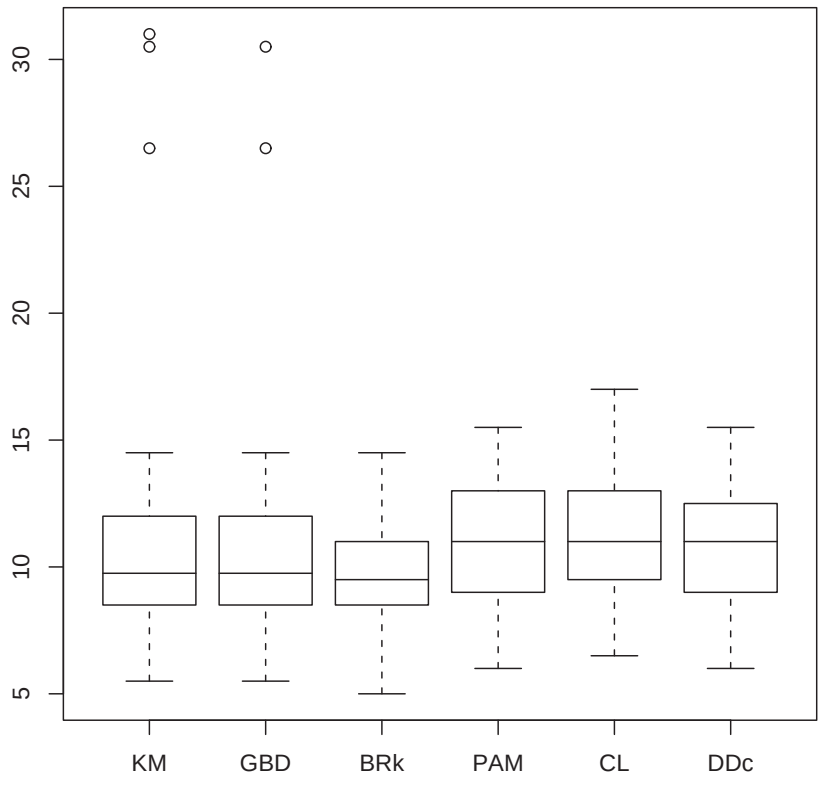


Figura 3.3: Diagramas de caja de las tasas de error obtenidas con los seis métodos para conjuntos de datos con 4 clusters distribuidos exponencialmente.

Método	Mínimo	Primer cuartil	Mediana	Tercer cuartil	Máximo
k-medias	5.50	**8.50**	9.75	12.00	31.00
GBDRk	5.50	8.50	9.75	12.00	30.50
BRk	**5.00**	**8.50**	**9.50**	**11.00**	**14.50**
PAM	6.00	9.12	11.00	12.87	15.50
CLARA	6.50	9.62	11.00	13.00	17.00
DDclust	6.00	9.00	11.00	12.50	15.50

Tabla 3.2: Valores mínimo, máximo y cuartiles de las tasas de error obtenidas para conjuntos de datos con 4 clusters distribuidos exponencialmente. ■

Ejemplo 3.3 *Tres clusters gaussianos multivariantes en dimensión 3, con medias linealmente dependientes y matriz de varianzas-covarianzas común, con el tercer grupo contaminado de forma asimétrica.*

Utilizamos este modelo descrito en Jörnsten *et al.* (2002), con medias $\mu_1 = (0,0,0), \mu_2 = (0,5,-5)$ y $\mu_3 = (0,-5,5)$, y matrices de varianzas-covarianzas

$$\Sigma_1 = \Sigma_2 = \Sigma_3 = \begin{pmatrix} 0.25 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Los clusters son independientes y tienen tamaños 25, 50 y 25, respectivamente. Contaminamos una proporción $\epsilon = 0{,}05$ de las observaciones del tercer grupo, con una distribución gaussiana

multivariante de media $\mu_{\mathcal{C}} = (0, 0, 0)$ y matriz de varianzas-covarianzas

$$\Sigma_{\mathcal{C}} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

Método	Mínimo	Primer cuartil	Mediana	Tercer cuartil	Máximo
k-medias	**0**	0.25	2	3	25
GBDRk	**0**	**0**	**1**	**2**	**4**
BRk	**0**	**0**	**1**	**2**	**4**
PAM	**0**	**0**	1.5	**2**	**4**
CLARA	**0**	**0**	**1**	**2**	**4**
DDclust	**0**	**0**	1.5	**2**	**4**

Tabla 3.3: Valores mínimo, máximo y cuartiles de las tasas de error obtenidas en tres grupos gaussianos con medias linealmente dependientes y un grupo contaminado.

En este modelo, los conjuntos están formados por clusters fácilmente separables, lo cual, cuando no hay contaminación, conduce a tasas de error cero en todos los métodos excepto en k-medias, que en ocasiones une dos clusters. Por tanto, todos los métodos tienen un comportamiento similar en los conjuntos contaminados, aunque es destacable que GBDRk mejora los resultados obtenidos por k-medias: proporciona cuartiles menores, lo cual refleja que en general GBDRk asigna correctamente una proporción de elementos mayor, y consigue dividir los clusters en los 4 casos en que habían sido incorrectamente unidos mediante k-medias, con sólo dos iteraciones. Los valores de la distorsión son muy similares en todos los métodos, difiriendo

únicamente en un conjunto (excepto en los cuatro casos en los que k-medias se comporta peor).

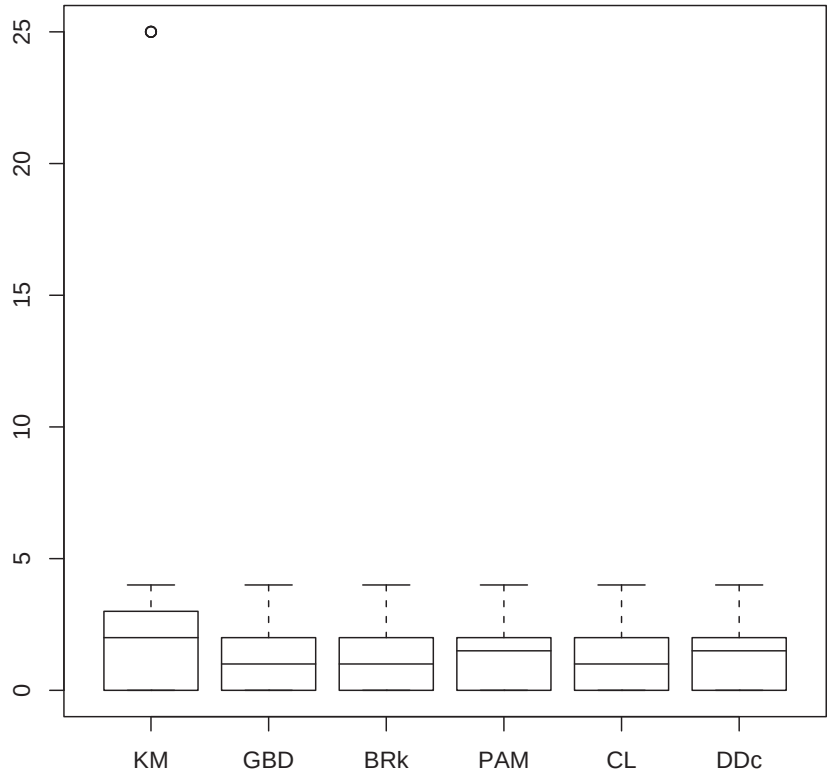


Figura 3.4: Diagramas de caja de las tasas de error obtenidas con los seis métodos para conjuntos con tres grupos gaussianos con medias linealmente dependientes y un cluster contaminado. ■

Ejemplo 3.4 *Tres clusters gaussianos multivariantes de dimensión 13, con 3 variables informativas y 10 variables ruido, con el segundo y el tercer grupos contaminados de forma asimétrica.*

Usamos el modelo descrito en Jörnsten (2004), con distribuciones normales multivariantes. Las medias para las variables informativas son $\mu_1 = (0, 0, 0)$, $\mu_2 = (\delta, -\delta, \delta)$ y $\mu_3 = (-\delta, -\delta, -\delta)$ y cero para las variables ruido. La matriz de varianzas-covarianzas para cada grupo es la matriz identidad en dimensión 13, I_{13}. Cada grupo tiene un tamaño de 50. A continuación mostramos los resultados para el caso $\delta = 2$.

Método	Mínimo	Primer cuartil	Mediana	Tercer cuartil	Máximo
k-medias	**4.00**	**8.00**	**10.67**	14.33	36.00
GBDRk	**4.00**	**8.00**	**10.67**	**13.33**	36.00
BRk	**4.00**	**8.00**	**10.67**	14.33	**21.33**
PAM	9.33	14.67	20.00	24.00	38.67
CLARA	6.67	16.00	20.00	26.67	38.67
DDclust	**4.00**	**8.00**	**10.67**	14.67	26.67

Tabla 3.4: Valores mínimo, máximo y cuartiles de las tasas de error obtenidas para 3 grupos gaussianos en dimensión 13, con 10 variables ruido, y dos clusters contaminados asimétricamente.

Cuando dos de los grupos se contaminan con una proporción $\epsilon = 0{,}05$ de observaciones procedentes de una distribución normal multivariante de media $\mu_{\mathcal{C}} = (0, \ldots, 0)$ y matriz de varianzas-covarianzas $\Sigma = 2 \times I_{13}$, el comportamiento de PAM y CLARA no es robusto, obteniendo un alto porcentaje de elementos asignados incorrectamente. En particular, para ambos métodos, la mediana de las distribuciones de dichos porcentajes es 20, (mayor que el peor de los valores obtenidos con GBDRk). Esto refleja la sensibilidad de PAM y CLARA a la

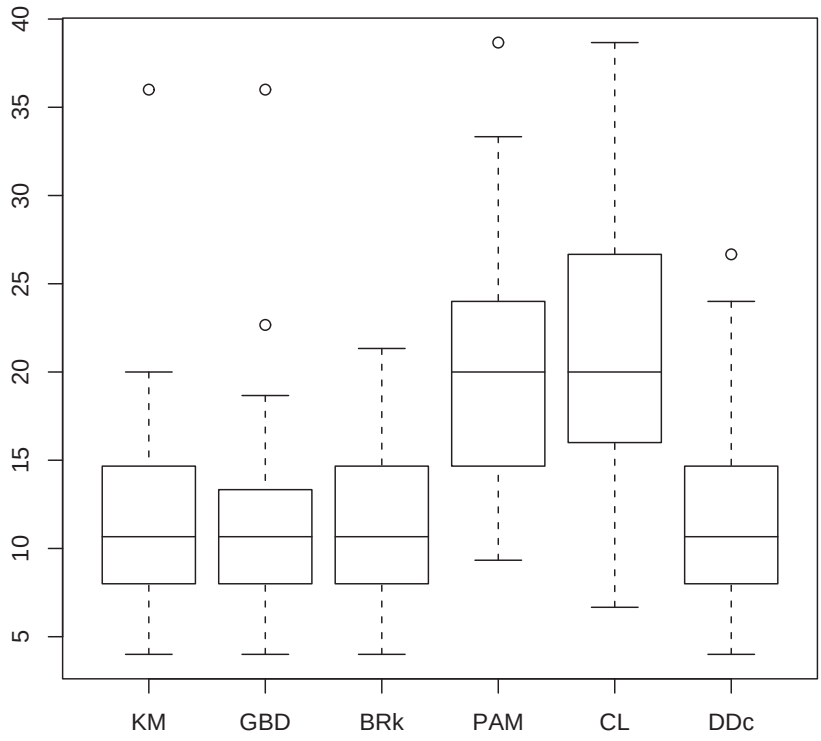


Figura 3.5: Diagramas de caja de las tasas de error obtenidas con los seis métodos para conjuntos con 3 grupos gaussianos en dimensión 13, con 10 variables ruido, y dos clusters contaminados de forma asimétrica.

inclusión de variables ruido. Para los otros cuatro métodos, este valor está alrededor de 11, lo cual muestra un comportamiento claramente más robusto. Observemos que k-medias no se ve muy afectada por las variables ruido debido a la convexidad de los clusters. El mejor valor del tercer cuartil corresponde a GBDRk, que alcanza tasas de error menores que las obtenidas por k-medias en 3 de los 50 casos, y en ningún caso mayores. Sin embargo, cuando los grupos son unidos de forma errónea con k-medias, GBDRk no los separa, mientras que BRk sí lo hace. Tanto GBDRk como BRk tienen un comportamiento similar al de DDclust. Con respecto a la

distorsión, BRk y GBDRk son de nuevo los mejores. ■

Ejemplo 3.5 *Tres clusters gaussianos multivariantes en dimensión 13, con 3 variables informativas y 10 variables ruido, con el segundo y tercer grupos contaminados simétricamente.*

Contaminamos de forma simétrica los clusters dos y tres del ejemplo 3.4 con distribuciones normales que tienen matriz de varianzas-covarianzas diagonal y varianzas iguales a 2. En este caso, aumentamos la proporción de elementos contaminados a $\epsilon = 0{,}1$. Los métodos menos robustos son PAM y CLARA (sensibles a la presencia de variables ruido), y presentan un primer cuartil de la distribución de la tasa de error mayor que el tercer cuartil en cualquiera de los otros métodos. En el caso en que GBDRk mejora a k-medias, se debe a que separa dos grupos erróneamente fusionados, pero en algunos casos no consigue hacerlo. Los mejores métodos son BRk y DDclust.

Método	Mínimo	Primer cuartil	Mediana	Tercer cuartil	Máximo
k-medias	**1.33**	6.67	9.33	10.67	22.67
GBDRk	**1.33**	6.67	8.67	10.67	17.33
BRk	**1.33**	**5.33**	**6.67**	8.67	**12.67**
PAM	5.33	12.33	16.00	20.00	42.67
CLARA	5.33	13.67	18.67	21.33	34.67
DDclust	2.67	6.67	7.33	**8.00**	41.33

Tabla 3.5: Valores mínimo, máximo y cuartiles de las tasas de error obtenidas para 3 grupos gaussianos en dimensión 13, con 10 variables ruido y dos clusters contaminados simétricamente.

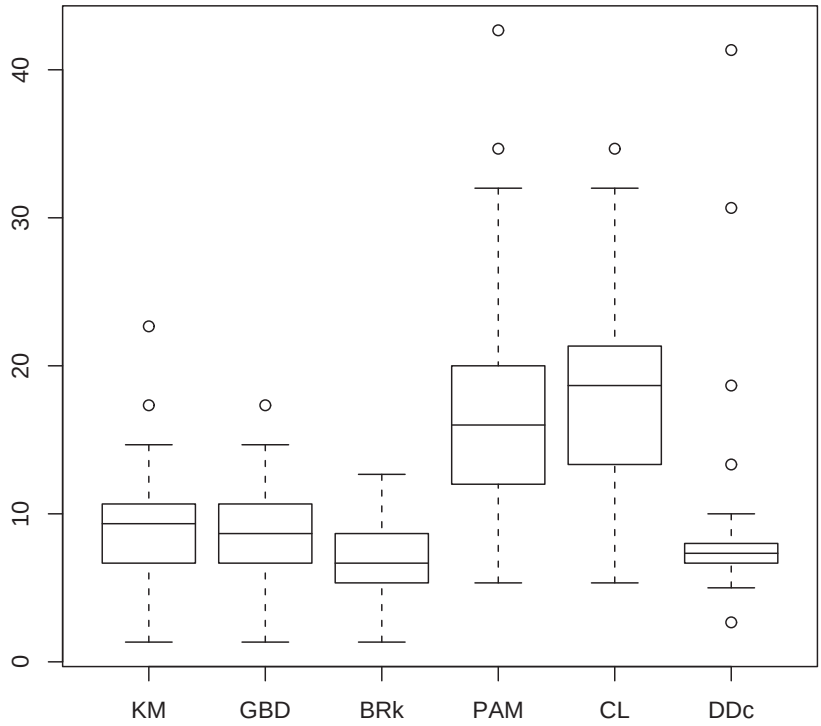


Figura 3.6: Diagramas de caja de las tasas de error obtenidas con los seis métodos para conjuntos con 3 grupos gaussianos en dimensión 13, con 10 variables ruido y dos clustes contaminados de forma simétrica.

El hecho más destacable del comportamiento de BRk es que, de nuevo, en todos los casos en los que k-medias une dos grupos de manera errónea, el método de refinamiento consigue separarlos, obviamente por ser independiente de los puntos iniciales que fueron escogidos en k-medias. ■

En resumen, podemos ver que en datos simulados, GBDRk produce mejores resultados que k-medias en un pequeño número de iteraciones, pero para algunos conjuntos de datos, sería interesante un comportamiento mejor, en el sentido de evitar los malos resultados en los que k-

medias queda atascada y de separar grupos que han sido unidos incorrectamente, para conseguir tasas de error menos variables. Por otra parte, BRk tiene en general un comportamiento muy robusto, similar e incluso mejor en ocasiones al de DDclust, siendo estos dos los mejores de todos los métodos utilizados. BRk consigue mejorar tanto la distorsión como la tasa de elementos mal asignados en k-medias, y obtiene sus resultados en mucho menos tiempo que DDclust.

3.3.2. Resultados en datos reales

Aplicamos los seis métodos al conjunto de datos de expresión génica descrito en Golub *et al.* (1999). Estos datos proceden de un estudio de las expresiones génicas en dos tipos de leucemia: leucemia linfoblástica aguda (ALL) y leucemia mieloide aguda (AML). El estudio fue realizado a partir de 25 casos de AML, y 47 casos de ALL (72 muestras en total). Los casos de ALL consistían en 38 muestras procedentes de células tipo B (ALL-B) y 9 de células de tipo T (ALL-T). Las expresiones génicas se midieron en 6817 genes (humanos), de forma simultánea. Redujimos el número de genes en estudio a 3571 como en Dudoit y Fridlyand (2002) siguiendo los siguientes pasos de pre-procesado. En primer lugar, se tomó una cota inferior de 100 y una cota superior de 16000, y todas las medidas que estaban por debajo de la cota inferior se sustituyeron por este valor, del mismo modo que las observaciones por encima de la cota superior se truncaron a dicha cota; a continuación, los datos se filtraron excluyendo aquellos genes que presentaban una relación señal-ruido baja, es decir, los que verificaban $max/min \leq 5$ o $(max - min) \leq 500$, donde max y min hacen referencia a las intensidades máxima y mínima, respectivamente, de cada gen en las 72 muestras. Finalmente, estandarizamos los datos de forma que cada muestra estuviese centrada en 0 y presentara varianza 1, para evitar que el análisis estuviera dominado por algún experimento en particular, como se sugiere en Jörnsten *et al.* (2002). La mayor parte de los genes en este conjunto no estaban activos, por lo que seleccionamos los 100 más variables (en las muestras) antes de continuar con el análisis. De

esta forma, el conjunto reducido consistía en $k = 3$ grupos de muestras en dimensión $d = 100$.

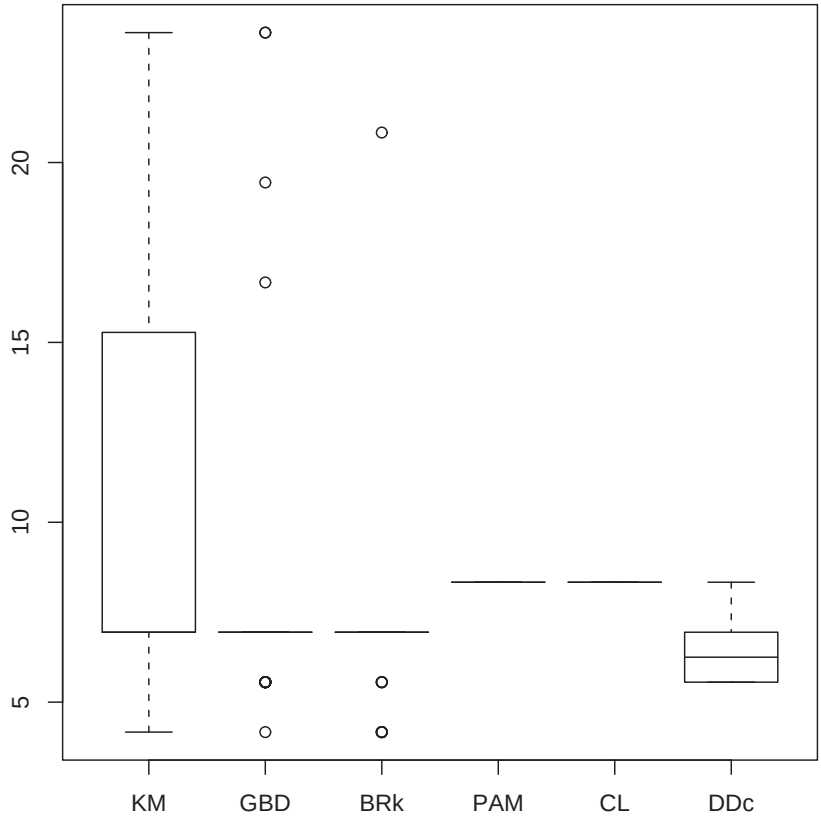


Figura 3.7: Diagramas de caja de las tasas de error obtenidas con los seis métodos en los datos de leucemia.

Aplicamos 50 veces cada uno de los métodos al conjunto resultante para agrupar las muestras en lugar de los genes; tanto PAM como CLARA produjeron siempre el mismo clustering, con una tasa de error de 8.33 %, que se corresponde con la asignación incorrecta de 6 muestras (en concreto, las etiquetadas con 28, 35, 38, 64, 66 y 67), de las cuales 5 corresponden a muestras AML y una a una muestra ALL-T. DDclust produjo clusterings distintos, con tasas de error entre 5.55 % y 8.33 % (que corresponde a la asignación incorrecta de entre 4 y 6 muestras). k-medias obtuvo resultados más variables, como se muestra en la figura 3.7, con la tasa de error

oscilando entre 4.167 % y 23.61 % (de 3 a 17 muestras). GBDRk mejoró notablemente estos resultados, haciendo caer la tasa de error en 9 de las 50 aplicaciones del método, y clasificando mal 5 muestras en 34 casos, y 4 muestras en 10 casos. Obsérvese que más de la mitad de las aplicaciones de k-medias produjeron un porcentaje de elementos mal asignados menor o igual que 6.944 % (5 muestras), lo cual permite a BRk formar, en media, clusters de centros compactos en los que buscar los puntos iniciales. Por tanto, los resultados de BRk son mucho más satisfactorios que los de k-medias, obteniendo un número de muestras mal asignadas mayor que 5 en sólo 4 de las 50 aplicaciones del método. Sin embargo, esto no es mucho mejor que lo obtenido por GBDRk, el cual asignaba mal más de 5 muestras en sólo 6 de los 50 casos. DDclust es el método que proporciona mejores resultados en este conjunto, con un valor máximo para la tasa de error significativamente más bajo que en los otros algoritmos.

Método	Mínimo	Primer cuartil	Mediana	Tercer cuartil	Máximo
k-medias	**4.167**	6.944	6.944	16.319	23.611
GBDRk	**4.167**	6.944	6.944	**6.944**	23.611
BRk	**4.167**	6.944	6.944	**6.944**	20.833
PAM	8.333	8.333	8.333	8.333	**8.333**
CLARA	8.333	8.333	8.333	8.333	**8.333**
DDclust	5.555	**5.555**	**6.249**	**6.944**	**8.333**

Tabla 3.6: Valores mínimo, máximo y cuartiles de las tasas de error obtenidas para los datos de leucemia.

Para estudiar el impacto producido por el número de variables incluidas en la precisión de los clusters, todos los procedimientos fueron aplicados a los datos de leucemia usando los $p = 500$ y 1000 genes con mayor varianza en las muestras. El comportamiento de PAM y

CLARA se deterioró al incluir variables ruido (genes no informativos), aumentando el número de muestras mal asignadas de 6 a 7 y 13, respectivamente, lo cual es consistente con los resultados obtenidos en nuestro estudio de simulación. Por otra parte, el comportamiento de los otros cuatro métodos permanece estable, siendo DDclust el más preciso, seguido por BRk y GBDRk. k-medias mostró la misma dispersión en la tasa de error que en el caso de $p = 100$ genes. ■

En resumen, en esta sección hemos propuesto dos métodos diferentes para encontrar las semillas iniciales del algoritmo de k-medias que mejoran considerablemente sus resultados. El primero, basado en la búsqueda del dato más profundo de cada cluster de los encontrados por k-medias, obtiene buenos resultados en general, pero en algunas ocasiones no es capaz de descartar mínimos locales en los que k-medias se queda atrapada. El segundo, que combina bootstrap con la búsqueda del dato más profundo de cada cluster (compacto) de centroides bootstrap, soluciona este problema, obteniendo resultados robustos que son comparables a los obtenidos por DDclust, otro método de clustering robusto y basado en profundidad de datos, pero con la ventaja de ser más rápido que éste.

El comportamiento del método de refinamiento basado en encontrar el dato más profundo de cada cluster bootstrap para formar el conjunto D, con $k \cdot B$ elementos, y en agrupar dicho conjunto en clusters, será objeto de investigación futura.

3.4. Clustering no rígido basado en bootstrap

Los algoritmos de clustering estándar que hemos usado hasta el momento asignan genes a clusters basándose en la similaridad de sus patrones de actividad. Genes con patrones similares deberían ser agrupados juntos, mientras que genes con distintos patrones de activación deberían ser colocados en clusters diferentes. Estos métodos de clustering están restringidos a correspondencias "uno-a-uno": un gen corresponde exactamente a un cluster. Mientras que este principio parece razonable en muchos campos del análisis cluster, puede resultar demasiado limitado para el estudio de datos procedentes de microarrays. Los genes pueden participar en diferentes redes genéticas y son coordinados frecuentemente por una gran variedad de mecanismos de regulación. Para el análisis de datos de microarrays, por tanto, podemos esperar que un gen pertenezca a varios clusters. Diversos investigadores han hecho notar que existen genes que frecuentemente están altamente correlados con múltiples grupos y que la definición de fronteras nítidas entre clusters de datos de expresiones génicas a menudo resulta arbitraria (Chu *et al.*, 1998; Cho *et al.*, 1998). Esto motiva el uso de técnicas tales como el algoritmo de c-medias difuso (FCM), introducido por Bezdek (1981).

En Teoría del Aprendizaje, el clustering difuso es una técnica que establece particiones en los datos, pero que permite pertenencias parciales, es decir, grados de pertenencia entre 0 y 1 en lugar de asignaciones rígidas de los datos a los clusters. Estas técnicas pertenecen a las técnicas de computación difusa, las cuales incluyen redes neuronales, sistemas difusos y algoritmos genéticos. En este sentido, el clustering no rígido que presentamos a continuación se puede considerar próximo a alguno de los algoritmos de este campo. Como ejemplo, véase Grotkjaer *et al.* (2006).

Consideramos que U es una partición no rígida del conjunto X si puede ser definida a través de una matriz de pertenencias cuyas celdas están confinadas en el intervalo unidad, y que satisface los dos requisitos siguientes (Pedrycz, 2005):

a) Los clusters son no triviales: para cada cluster obtenemos una estructura en la matriz que no incluye todos los datos.

b) Los grados de pertenencia totales suman 1.

El método presentado en la sección 3.2.3 permite la construcción de una partición no rígida de los datos que en ocasiones puede ser más apropiada para agrupar un conjunto dado. Consideremos el conjunto de centros $\mathcal{C} = \{\mathcal{C}_i^b, i = 1, ..., k, b = 1, ..., B\}$ obtenido en el paso 3 del algoritmo BRk, que es agrupado para obtener k clusters $I_1^*, ..., I_k^*$. Podemos construir una región de confianza para las semillas iniciales, que recordemos que se calculaban como los puntos más profundos $\mathcal{D}_i$ en cada cluster I_i^*, eliminando primero los puntos con menor profundidad hasta que se alcanza el tamaño deseado, y formando a continuación la envolvente convexa de los puntos restantes como se describe en Yeh y Singh (1997). Denotaremos por $W_{1-\alpha}^*(\mathcal{D}_i)$ la región obtenida al eliminar los $100\alpha\,\%$ puntos bootstrap más externos de I_i^*, usando la profundidad por bandas generalizada, y por $H(A)$ la envolvente convexa de un conjunto A.

Consideremos k puntos distintos en $\mathcal{R}^d$, $y_1, \ldots, y_k$. Las **regiones de Voronoi** de los puntos $y_1, \ldots, y_k$ (ver, por ejemplo, de Berg *et al.* (2000)), o los **dominios de atracción** para los puntos $y_i, i = 1, ..., k$, se definen como:

$$\mathcal{A}(y_i) = \{x \in \mathcal{R}^d : d(x, y_i) < d(x, y_j), i \neq j\}.$$

Las regiones de Voronoi de un conjunto de k puntos distintos crean una partición con k regiones o celdas convexas de $\mathcal{R}^d$. El algoritmo de k-medias consiste en encontrar las regiones de Voronoi de los k centroides. Véase la figura 3.8.a). Obsérvese que los puntos que se encuentran en las fronteras de las celdas se asignan arbitrariamente a uno de los clusters adyacentes.

a) b)

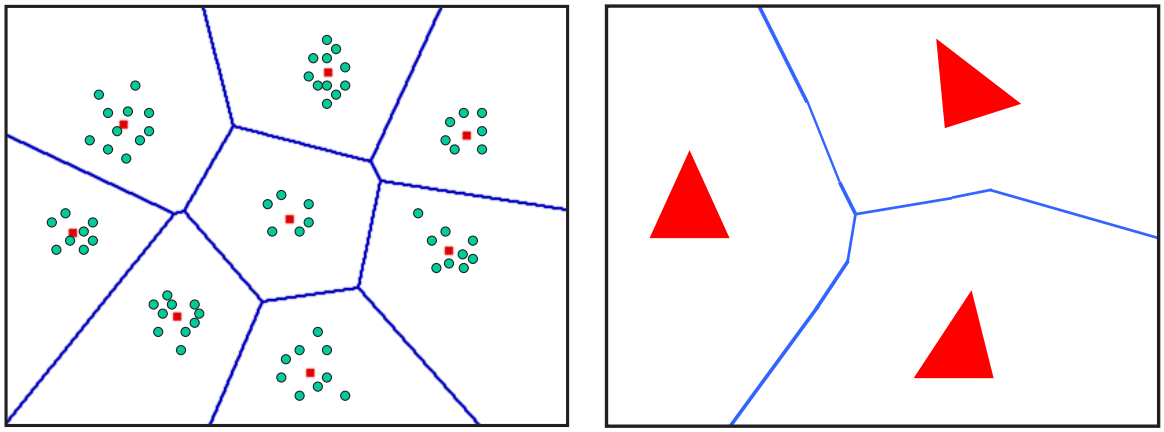


Figura 3.8: a) Ocho clusters con los correspondientes centros (medias) dibujados en rojo. Las regiones de Voronoi convexas (polígonos) están delimitadas por líneas azules. b) Regiones de Voronoi generalizadas no convexas relativas a los dominios de atracción de tres triángulos distintos.

Consideremos los k puntos más profundos $\mathcal{D}_i$ obtenidos en el método BRk, asumiendo que son distintos. El dominio de atracción del punto $\mathcal{D}_i$ se puede generalizar a otros lugares geométricos, y en particular, el dominio de atracción de la envolvente convexa del conjunto $W^*_{1-\alpha}(\mathcal{D}_i)$ está dado por

$$\mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_i))) = \{x \in \mathcal{R}^d : d(x, H(W^*_{1-\alpha}(\mathcal{D}_i))) < d(x, \mathcal{D}_j), i \neq j\},$$

donde la distancia de un punto x a un conjunto A está definida por

$$d(x, A) = min_{a \in A} d(x, a).$$

Esto determina una partición de Voronoi generalizada del espacio, con regiones que en general no son convexas, y que conduce por tanto a una agrupación diferente de los datos de la dada por k-medias.

Proposición 3.2 *El dominio de atracción generalizado de la envolvente convexa de* $W^*_{1-\alpha}(\mathcal{D}_i)$ *contiene las regiones de Voronoi de* $\mathcal{D}_i$.

Demostración: Es inmediato teniendo en cuenta que $\mathcal{D}_i$ es el punto más profundo de I^*_i y por tanto, para cada α, $\mathcal{D}_i \in W^*_{1-\alpha}(\mathcal{D}_i)$. De aquí, si $z \in \mathcal{A}(\mathcal{D}_i)$, $d(z, \mathcal{D}_i) < d(z, \mathcal{D}_j), j \neq i$ y por tanto $d(z, H(W^*_{1-\alpha}(\mathcal{D}_i))) \leq d(z, \mathcal{D}_i) < d(z, \mathcal{D}_j), j \neq i$. ■

La proposición 3.2 justifica el siguiente método para generar un clustering no rígido del conjunto X. Construiremos las regiones de Voronoi generalizadas de los lugares geométricos (distintos) $\mathcal{D}_1, \ldots, \mathcal{D}_{i-1}, W^*_{1-\alpha}(\mathcal{D}_i), \mathcal{D}_{i+1}, \ldots, \mathcal{D}_k$, para cada $i = 1, ..., k$. La intersección de las k particiones origina $2^k - 1$ regiones disjuntas, que en general no son convexas, algunas de las cuales pueden ser vacías, y que se pueden describir como sigue:

$$\begin{aligned} I^{(k)} &= \bigcap_{i=1}^{k} \mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_i))) \\ I_j^{(k-1)} &= \bigcap_{i \neq j} \mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_i))) - I^{(k)}, j = 1, ..., k \end{aligned}$$

$$I^{(k-2)}_{j_1,j_2} = \bigcap_{i\neq j_1,j_2} \mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_i))) - I^{(k)} - \bigcup_{j=1}^{k} I^{(k-1)}_j, j_1 \neq j_2$$

$$\vdots$$

$$I^{(1)}_{j_1,...,j_{k-1}} = \mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_{j_k}))) - I^{(k)} - ... - \bigcup I^{(2)}_{j_1,...,j_{k-1}}, j_k = 1, ..., k$$

En ocasiones nos referiremos genéricamente a estas regiones sin mencionar los subíndices. Los puntos que pertenecen a $I^{(1)}_{j_1,...,j_{k-1}}$ son los que no han sido atraídos por ningún otro centro o su correspondiente región de confianza aparte de D_{j_k}. En la figura 3.9 podemos ver las 7 regiones, $I^{(3)}$, $I^{(2)}_1$, $I^{(2)}_2$, $I^{(2)}_3$, $I^{(1)}_{1,2}$, $I^{(1)}_{1,3}$ y $I^{(1)}_{2,3}$, originadas al calcular las tres particiones de Voronoi generalizadas correspondientes a dos puntos y un triángulo (que representa la envolvente convexa de la región de confianza del tercer punto). Estas regiones están limitadas por líneas azules discontinuas. Las líneas negras continuas son las fronteras de los dominios de atracción de los puntos $\mathcal{D}_1$, $\mathcal{D}_2$ y $\mathcal{D}_3$.

Las regiones de Voronoi de los puntos $\mathcal{D}_1, \ldots, \mathcal{D}_k$ determinan una partición $C_1, \ldots, C_k$ del conjunto X que puede ser descrita en términos de probabilidades como sigue:

$$P_0\{x \in C_i\} = \begin{cases} 1, & \text{si } x \in \mathcal{A}(\mathcal{D}_i) \\ \frac{1}{h} & \text{si } x \in \delta(\mathcal{A}(\mathcal{D}_i)) \\ 0, & \text{en otro caso,} \end{cases}$$

donde $\delta(A)$ representa la frontera del conjunto A y h es el número de celdas de Voronoi cuyas fronteras incluyen al punto x.

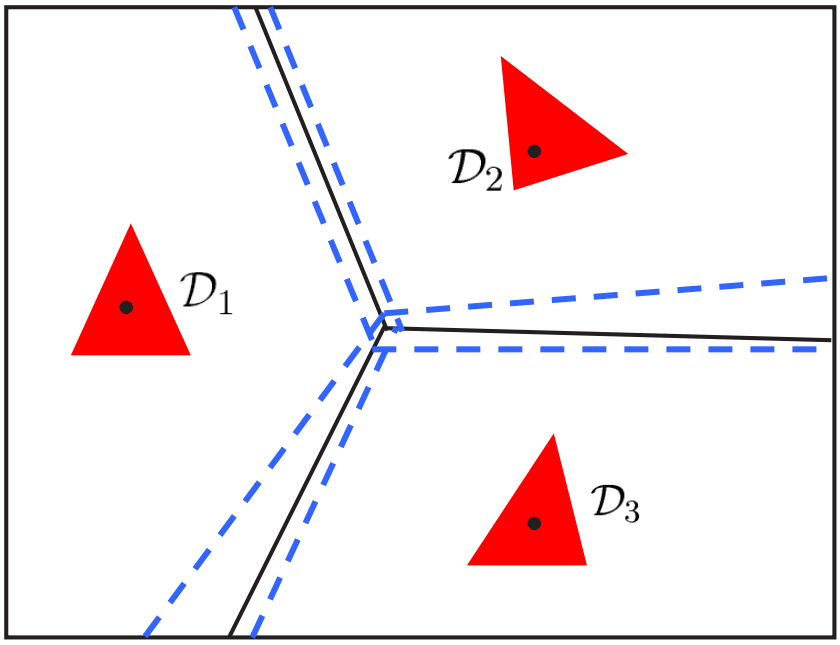


Figura 3.9: Siete regiones disjuntas correspondientes a las tres particiones de Voronoi generalizadas para dos puntos y un triángulo. Las fronteras de estas regiones son las líneas azules discontinuas. Las fronteras de las regiones de Voronoi "estándar" correspondientes a los tres puntos están representadas por líneas negras continuas.

Para un tamaño dado $100(1-\alpha)\,\%$ de las regiones de confianza bootstrap, podemos definir una partición no rígida del conjunto X en clusters $F_1, \ldots, F_k$, generada por los centros $\mathcal{D}_1$,...,$\mathcal{D}_k$, y descrita por las siguientes probabilidades: primero, si x está en la frontera de algún $\mathcal{A}(D_i)$ entonces es asignado arbitrariamente a cualquiera de los clusters adyacentes; segundo, si x está en cualquiera de las k regiones $I^{(1)}$, entonces la probabilidad de pertenecer al cluster F_i

está dada por:

$$P_\alpha\{x \in F_i\} = P_0\{x \in C_i\}$$

y, finalmente, si x está en cualquiera de las otras regiones $I^{(q)}_{j_1,...,j_{k-q}} - \cup_i \delta(\mathcal{A}(\mathcal{D}_i))$, $q > 1$, la probabilidad de pertenecer al cluster F_i está dada por:

$$P_\alpha\{x \in F_i\} = \begin{cases} \dfrac{1}{q} + \dfrac{\alpha}{q}, & \text{si } x \in \mathcal{A}(\mathcal{D}_i), i \neq j_1, ..., j_{k-q} \\ \dfrac{1}{q} - \dfrac{\alpha}{q(q-1)}, & \text{si } x \notin \mathcal{A}(\mathcal{D}_i), i \neq j_1, ...j_{k-q} \\ 0, & \text{en otro caso.} \end{cases} \tag{3.4.3}$$

Así, si un punto no es atraído por otro centro o región de confianza aparte de $\mathcal{D}_i$, entonces será asignado al cluster F_i con probabilidad 1. La probabilidad de asignación de los restantes puntos tiene en cuenta a las regiones de confianza que los atraen. A los puntos en las fronteras se les asigna arbitrariamente la probabilidad de pertenencia correspondiente a cualquiera de las regiones adyacentes.

$P_\alpha(.)$ y, en particular, la función de probabilidad (3.4.3) está bien definida, como se demuestra a continuación.

Proposición 3.3 *Si un punto* $x \in I^{(q)}_{j_1,...,j_{k-q}}$ *entonces* $x \notin \mathcal{A}(\mathcal{D}_j), j = j_1, ..., j_{k-q}$.

Demostración: En efecto, si $x \in I^{(q)}_{j_1,...,j_{k-q}}$ y $x \in \mathcal{A}(\mathcal{D}_g)$, para algún $g \in \{j_1, ..., j_{k-q}\}$,

entonces $x \in \mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_g)))$, y $x \in \left(\bigcap_{i \neq j_1,...,j_{k-q}} \mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_i))) \cap \mathcal{A}(H(W^*_{1-\alpha}(\mathcal{D}_g))) \right)$.

Por tanto, existe un entero positivo l tal que $x \in I^{(q+l)}$ y $x \notin I^{(q)}_{j_1,...,j_{k-q}}$, lo cual no es posible. Así, $x \notin \mathcal{A}(\mathcal{D}_g)$, $\forall g \in \{j_1, ..., j_{k-q}\}$. ■

Por consiguiente, si $x \notin \delta(\mathcal{A}(\mathcal{D}_i))$ para algún i, entonces pertenece a algún $\mathcal{A}(\mathcal{D}_i)$, $i \neq j_1, ..., j_{k-q}$, y (3.4.3) está bien definida.

Proposición 3.4 *Los grados de pertenencia dados por $P_\alpha\{x \in F_i\}$ suman 1.*

Demostración: Trivialmente, si $x \in I^{(1)}$ o $x \in \delta(\mathcal{A}(D_j))$ para algún j, entonces $\sum_{i=1}^{k} P_\alpha\{x \in F_i\} = P_0\{x \in C_i\} = 1$.

Por otra parte, si $x \in I^{(q)}_{j_1,...,j_{k-q}} - \cup_i \delta(\mathcal{A}(\mathcal{D}_i))$, $q > 1$, entonces $\exists s \neq j_1, ..., j_{k-q}$ tal que $x \in \mathcal{A}(\mathcal{D}_s)$. Por tanto,

$$\sum_{i=1}^{k} P_\alpha\{x \in F_i\} = \sum_{i=j_1,...,j_{k-q}} P_\alpha\{x \in F_i\} + \sum_{i \neq j_1,...,j_{k-q}} P_\alpha\{x \in F_i\} = 0 + P_\alpha\{x \in \mathcal{A}(\mathcal{D}_s)\} +$$

$$+ \sum_{i \neq j_1,...,j_{k-q},s} P_\alpha\{x \in \mathcal{A}(\mathcal{D}_i)\} = \frac{1}{q} + \frac{\alpha}{q} + (q-1)\left(\frac{1}{q} - \frac{\alpha}{q(q-1)}\right) = \frac{q}{q} = 1. \quad ■$$

La figura 3.10 muestra las probabilidades de asignación de tres clusters no rígidos, codificados por colores. Las probabilidades de los puntos en las fronteras de las regiones de Voronoi no se muestran.

El inconveniente práctico de este método es el coste computacional de las regiones de Voronoi para datos de alta dimensión. Sin embargo, existen algoritmos aproximados que calculan tales

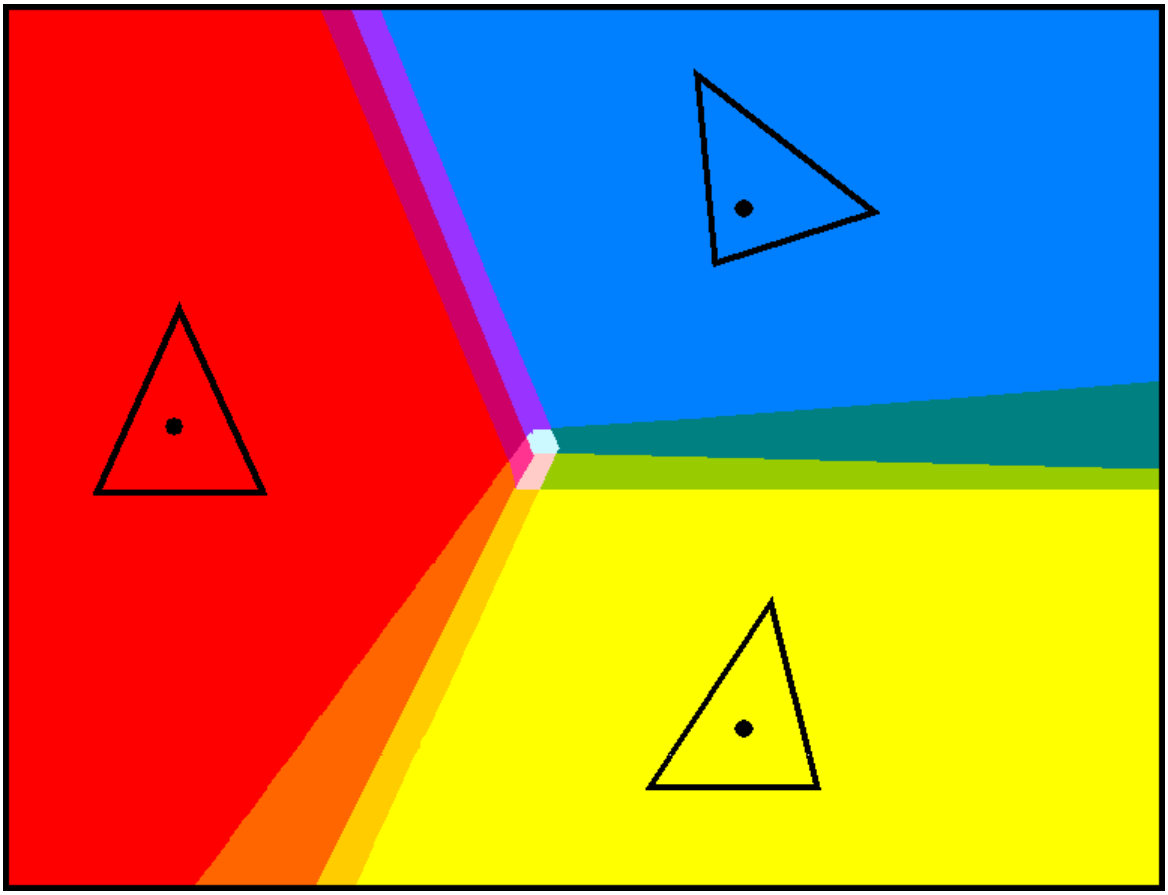


Figura 3.10: Probabilidades de asignación de un punto x a cada uno de los tres clusters no rígidos, codificados por colores. Cuanto más próximo esté el color correspondiente al rojo, azul o amarillo puros, mayor será la probabilidad de pertenecer a uno de estos clusters.

regiones y que pueden ser utilizados para acelerar los cálculos. Véase, por ejemplo, Hoff *et al.* (1999) y las referencias contenidas en este trabajo.

3.4.1. Resultados en los datos de leucemia

Comprobamos el método propuesto en los datos de leucemia para determinar un clustering no rígido de las 72 muestras. Considerando tres posibles clusters, ALL-B, ALL-T y AML, podemos construir 7 regiones de Voronoi generalizadas (aproximadas) en dimensión $d = 100$. Tras asignar las 72 muestras en estas regiones, encontramos que 49 de las 72 muestras pertenecían a una de las regiones $I^{(1)}$, y que todas ellas estaban bien clasificadas; 12 muestras pertenecían a algunas de las regiones $I^{(2)}$ y 11 de ellas pertenecían a $I^{(3)}$. La distribución del número de muestras en las siete regiones del espacio 100-dimensional está dada en la tabla 3.7. Las 5 muestras que habían sido mal clasificadas con el método BRk (muestras 28, 35, 38 y 66, de tipo AML y la muestra 67, de tipo ALL-T) pertenecen a las regiones $I^{(3)}$. Las probabilidades de pertenencia dadas por el método se recogen en la tabla 3.8.

Tipo de tumor	$I^{(1)}_{2,3}$	$I^{(1)}_{1,3}$	$I^{(1)}_{1,2}$	$I^{(2)}_{3}$	$I^{(2)}_{2}$	$I^{(2)}_{1}$	$I^{(3)}$
ALL-B	28	0	0	6	2	0	2
ALL-T	0	6	0	0	0	0	3
AML	0	0	15	3	0	1	6

Tabla 3.7: Distribución de las muestras en las regiones de Voronoi.

Este conjunto de datos reales muestra que incluso si algunas muestras no se asignan de forma única a un cluster, el método es capaz de identificar aquellas muestras que habían sido mal clasificadas usando un algoritmo de clustering estándar, y de asignarle una probabilidad balanceada de pertenecer a cada cluster.

Probabilidad	ALL-B	ALL-T	AML	Grupo real
Muestra 28	0.35	0.325	0.325	AML
Muestra 35	0.35	0.325	0.325	AML
Muestra 38	0.325	0.35	0.325	AML
Muestra 66	0.35	0.325	0.325	AML
Muestra 67	0.35	0.325	0.325	ALL-T

Tabla 3.8: Probabilidades de pertenencia a cada uno de los clusters formados por las muestras de los tres tipos de tumor. ■

Capítulo 4

Estimación del número de grupos

Resumen

Como se mencionó previamente, un aspecto esencial del problema de clustering que debería ser abordado antes de utilizar un método particular en un conjunto de datos dado es la estimación del número de grupos subyacentes, si existe alguno. En este capítulo presentamos un estudio diseñado para estimar el número de grupos en la estructura de los datos, aprovechando el algoritmo greedy utilizado en el método de comparación de clusterings, descrito en la sección 2.2.1, y la definición de profundidad empleada en el capítulo anterior. En el primer caso, dependiendo del número de clusters usados en cada comparación, proponemos dos enfoques, el k-único (single k) y el k-mixto (mixed k). En el segundo caso, usamos ideas previas acerca del diámetro de un conjunto, combinadas con la noción de profundidad. Encontramos que el método k-único proporciona una estimación razonable del rango en el cual utilizar el método k-mixto, y para una gran variedad de conjuntos de datos, la estimación basada en los diámetros y en la profundidad proporciona un valor preciso del número de grupos.

4.1. Introducción

Como se mencionó anteriormente, el clustering es una técnica de aprendizaje no supervisado en el cual se trata de encontrar una serie de grupos sin la ayuda de una variable respuesta. Uno de los mayores retos en el análisis cluster es la estimación del número de grupos en un conjunto dado, lo cual es esencial para realizar una agrupación de los datos efectiva y eficiente. Por ejemplo, un algoritmo de clustering como k-medias puede generar una mala agrupación si las particiones iniciales no son escogidas de forma apropiada, una situación que ocurre a menudo y cuya solución no es una tarea trivial.

Se han propuesto muchos métodos para estimar el número de grupos. Como se dijo en la introducción, algunas propuestas que se han usado en datos de expresión génica, o que han sido expresamente diseñadas para este tipo de datos son las siguientes:

- el **estadístico "silueta"**, propuesto en Kaufman y Rousseeuw (1990). Dada una observación i sea $a(i)$ la distancia media de este punto a las otras observaciones del mismo cluster, y sea $b(i)$ la distancia media del punto a las observaciones del cluster más próximo (descartando el propio). Entonces, se define el estadístico silueta como

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}.$$

  Un punto está bien asignado si $s(i)$ toma un valor alto. Los autores proponen escoger como número óptimo de clusters el valor que maximiza la media de los $s(i)$ en el conjunto completo, $sil = \sum_i \frac{s(i)}{n}$.

  Obsérvese que $s(i)$ no está definido para $k = 1$ cluster.

- el **estadístico gap**, propuesto en Tibshirani *et al.* (2001). Este método compara la suma de cuadrados intra-clusters con su esperanza bajo una distribución nula de referencia como se explica a continuación. Para cada número posible de clusters $k \geq 1$, se calcula la suma de cuadrados intra clusters trW_k. Se generan B conjuntos de referencia bajo la distribución nula y se aplica el algoritmo de clustering a cada uno de ellos, calculando las correspondientes sumas de cuadrados intra-clusters trW_k^1,...,trW_k^B. Finalmente, se calcula el estadístico gap estimado como:

$$gap(k) = \frac{1}{B}\sum_b logtrW_k^b - logtrW_k$$

y la desviación estándar sd_k de trW_k^b, $1 \leq b \leq B$. Sea $\widetilde{sd_k} = sd_k\sqrt{1+1/B}$. El número de clusters estimado es el menor $k \geq 1$ tal que

$$gap(k) \geq gap(k+1) - \widetilde{sd_{k+1}}.$$

Tibshirani *et al.* eligieron la hipótesis de uniformidad para crear la distribución nula de referencia, y consideraron dos posibles maneras de construir la región de soporte de tal distribución. En la primera posibilidad, la ventana de muestreo para la j-ésima variable ($1 \leq j \leq d$) se limita al rango de los valores observados en esta variable. En la segunda opción, las variables se muestrean a partir de una distribución uniforme confinada a una "caja" alineada con las componentes principales de la matriz del diseño, centrada. En detalle, si X es la matriz de datos, supongamos que las columnas tienen media cero y calculemos la descomposición en valores singulares $X = UDV^T$. Transformamos via $X' = XV$ y a continuación obtenemos las variables uniformes Z' en los rangos de las correspondientes columnas de X'. Finalmente, se deshace la transformación en la nueva matriz de diseño vía

$Z = Z'V^T$ para obtener el conjunto de referencia. Este segundo enfoque se denota por $gapPC$, donde PC corresponde a las Componentes Principales. Obsérvese que, en este caso, el método sí permite la estimación de $k = 1$ cluster.

- el **método CLEST**, introducido por Dudoit y Fridlyand (2002). El algoritmo consiste en dividir repetidamente la muestra en un conjunto de entrenamiento L^b y un conjunto de validación T^b. Para cada iteración y para cada posible número de clusters k se obtiene un clustering $P(\cdot, L^b)$ del conjunto de entrenamiento y se construye un predictor $C(\cdot, T^b)$ usando las etiquetas de las clases obtenidas en el clustering. A continuación, se aplica el predictor al conjunto de validación y las etiquetas predichas se comparan con las obtenidas al agrupar (con algún método de clustering) los datos del conjunto de validación, usando algún estadístico de similaridad (por ejemplo, el índice de Rand). El número de clusters se estima comparando el estadístico de similaridad observado para cada k con su valor esperado bajo alguna distribución nula apropiada con $k = 1$. El número estimado de clusters se define como el $\hat{k}$ que proporciona mayor evidencia en contra de la hipótesis nula.

- el **estadístico ReD**, propuesto por Jörnsten (2004) y basado en la noción de profundidad. Es muy similar al estadístico silueta. Para una observación i, ReD_i es la diferencia entre las profundidades con respecto al cluster en el cual ha sido asignada, denotada por D_i^w, y el cluster más próximo, denotado por D_i^b. El número de clusters seleccionado por el estadístico ReD es el valor $\hat{k}$ que maximiza la profundidad relativa media $ReD = \sum_i \eta_i ReD_i$, donde η_i es la multiplicidad de la observación i. A diferencia de sil, ReD es independiente de las escalas de los clusters, por lo que

no está dominado por clusters de gran varianza.

En este capítulo proponemos dos métodos para estimar el número de grupos de un conjunto; el primero toma ideas del método de comparación de clusterings descrito en el capítulo 2; el segundo es un método muy simple, basado en seleccionar subconjuntos de cada cluster, o más precisamente, los $\alpha\,\%$ puntos más profundos en el clusters, y en calcular la suma de los diámetros de tales subconjuntos. De nuevo, la noción de profundidad que usaremos en este capítulo es la profundidad por bandas generalizadas descrita en López-Pintado y Romo (2006).

4.2. Comparación de pares de clusterings

El algoritmo greedy propuesto en el capítulo 2 conduce de manera natural al problema de estudiar la distribución del número de súper-clusters que resultan al comparar un par de clusterings del mismo conjunto de datos. La dificultad de este estudio, que deriva de la ausencia general de resultados distribucionales relacionados con los algoritmos de clustering, y en particular con k-medias, sugiere el uso de técnicas bootstrap.

Para un conjunto dado X con N elementos, extraído de una población con k clusters, es posible encontrar la distribución bootstrap del número de súper-clusters p, suponiendo que hemos aplicado dos veces k-medias a X, iniciando con k clusters, en los pasos siguientes:

1. FOR i in 1:B DO

   1.1 Obtener una muestra bootstrap X_i^* de X con N elementos.

   1.2 Aplicar k-medias dos veces a X_i^* para obtener los clusterings $\mathbf{A_i^*}$ y $\mathbf{B_i^*}$, cada uno con k grupos.

1.3 Usar el algoritmo greedy para calcular el número de súper-clusters p_i^*.

2. Calcular la distribución bootstrap del número de súper-clusters p.

3. Estimar p con la moda de esta distribución bootstrap.

Denotemos a los centros de los k grupos de la población de la cual ha sido extraído X por $\mu_1, \ldots, \mu_k$. Supongamos que aplicamos k-medias con k clusters a X y obtenemos el clustering $\mathbf{A} = \{A_1, \ldots, A_k\}$, y denotemos a los correspondientes centros de los clusters por $\{a_1, \ldots, a_k\}$. Si se verifican las condiciones del teorema de Pollard, expuesto en Pollard (1982), entonces los centros muestrales convergen a los centros poblacionales $\mu_1, \ldots, \mu_k$, salvo por reordenación de los índices, y por tanto, cuando el tamaño muestral es suficientemente grande, el número de súper-clusters converge al número real de grupos, k.

Desafortunadamente, este razonamiento falla en la práctica, pues habitualmente el número de grupos es desconocido, y además tampoco se conocen resultados distribucionales teóricos para k-medias cuando el número de clusters utilizado al aplicar el método es diferente del k de la población; por supuesto, el mismo procedimiento anterior puede ser utilizado para calcular la distribución bootstrap del número de súper-cluster para dos clusterings dados con un número de clusters diferente al de la población k, pero no podemos esperar que p converja a k. Así, en los casos prácticos en los que el número de grupos k es desconocido, no podremos utilizar p como un estimador preciso de k. Para ilustrar cómo se incrementa el valor de p cuando variamos el número de clusters utilizado en k-medias, consideremos las siguientes simulaciones.

Ejemplo 4.1 *Modelo 1.*

Generamos diversos conjuntos de datos según el modelo 1, con 4 grupos, nivel de ruido $\sigma = 4$ y distintos tamaños para los clusters $c_1 = c_2 = c_3 = c_4 = i$, siendo $i = 25, 50, 100, 200$ y 500, y a continuación calculamos la distribución bootstrap de p al usar k-medias con 4, 5, 6 y 7 clusters.

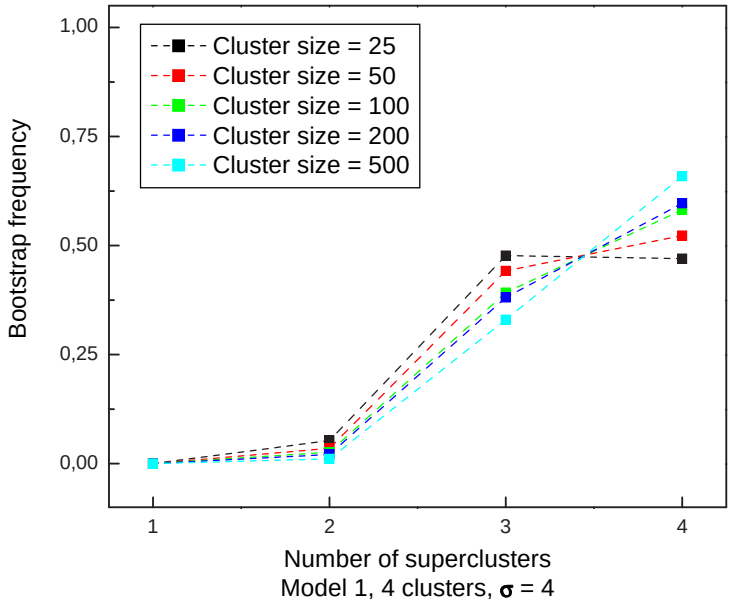


Figura 4.1: Modelo 1 con 4 clusters, $\sigma = 4$ y diferentes tamaños de los clusters. Distribución bootstrap de p cuando se aplica k-medias con 4 clusters.

Para 4 clusters, se observa claramente que, al aumentar el tamaño de los clusters, la frecuencia de los 4 super-clusters aumenta también, tal y como esperábamos. Para 5 clusters, la distribución bootstrap de p tiene también una clara moda en 4 para todos los tamaños de los clusters, pero no es el caso de 6 y 7 clusters. Para 6 clusters y los tamaños de clusters más pequeños, la moda sigue siendo 4, pero para tamaños a partir de 200 es igualmente probable encontrar 4 o 5 super-clusters. Finalmente, para 7 clusters, la moda es 5. Aunque no mostramos los resultados aquí, hemos observado que a medida que el número de clusters aumenta, la moda de la distribución bootstrap se aleja cada vez más de 4.

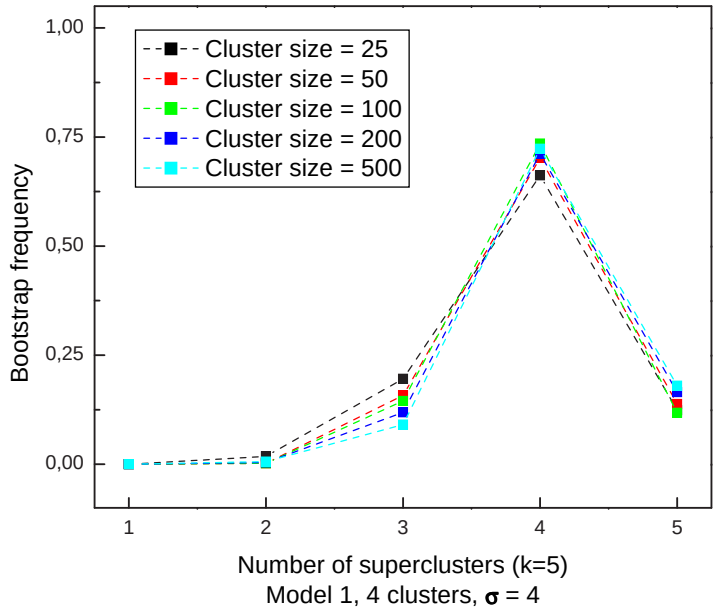


Figura 4.2: Modelo 1 con 4 clusters, $\sigma = 4$ y diferentes tamaños de los clusters. Distribución bootstrap de p cuando se aplica k-medias con 5 clusters.

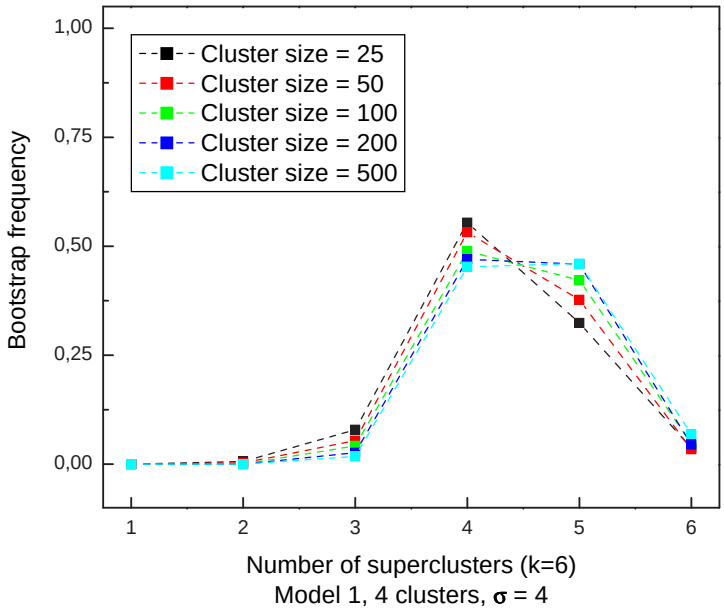


Figura 4.3: Modelo 1 con 4 clusters, $\sigma = 4$ y diferentes tamaños de los clusters. Distribución bootstrap de p cuando se aplica k-medias con 6 clusters.

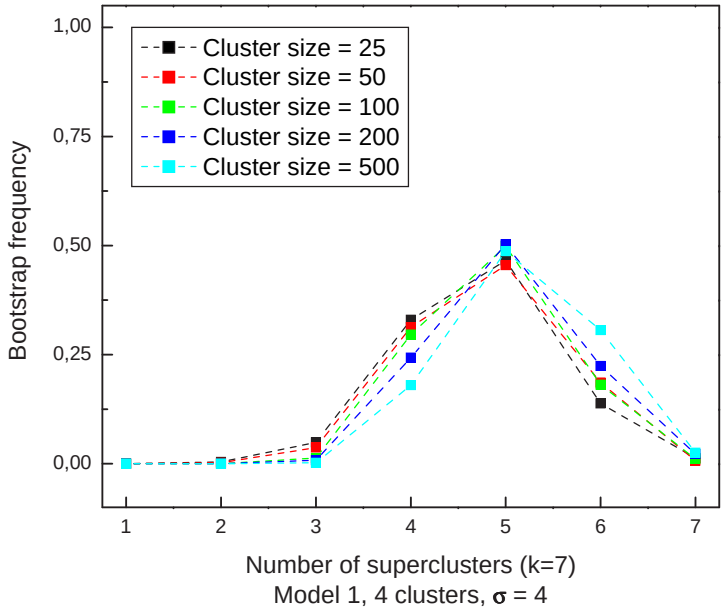


Figura 4.4: Modelo 1 con 4 clusters, $\sigma = 4$ y diferentes tamaños de los clusters. Distribución bootstrap de p cuando se aplica k-medias con 7 clusters. ■

Ejemplo 4.2 *Modelo 2.*

Usamos conjuntos generados según el modelo 2, tomando como tamaños de los clusters $c_1 = c_2 = \frac{1}{2}c_3 = i$, siendo $i = 25, 50, 100, 200$ y 500, y a continuación calculamos la distribución bootstrap de p al aplicar k-medias con 3, 4, 5 y 6 clusters.

Como ocurre en el ejemplo anterior, en los casos en los que aplicamos k-medias con 3 y 4 clusters, la distribución bootstrap del número de súper-clusters tiene una sola moda en 3, el verdadero número de grupos; sin embargo, cuando el número de clusters aumenta, la moda también aumenta, como se puede observar en las figuras 4.5, 4.6, 4.7 y 4.8.

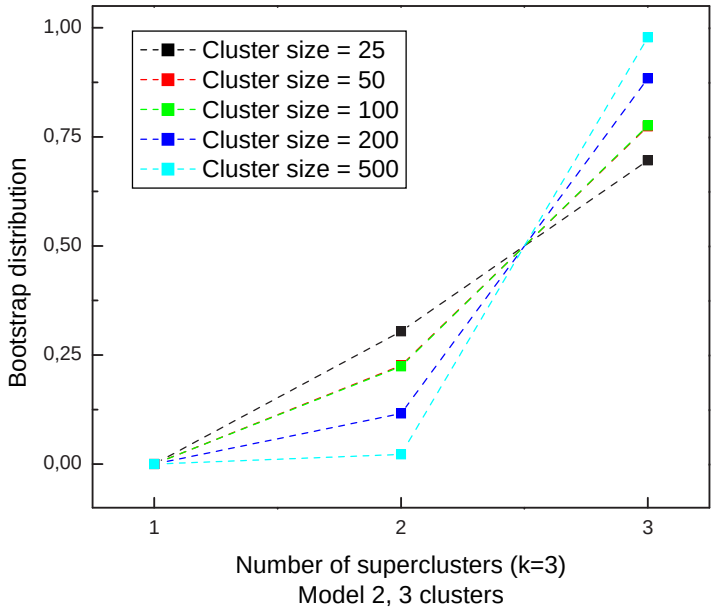


Figura 4.5: Modelo 2 con 3 clusters y diferentes tamaños de los clusters. Distribución bootstrap de p al aplicar k-medias con 3 clusters.

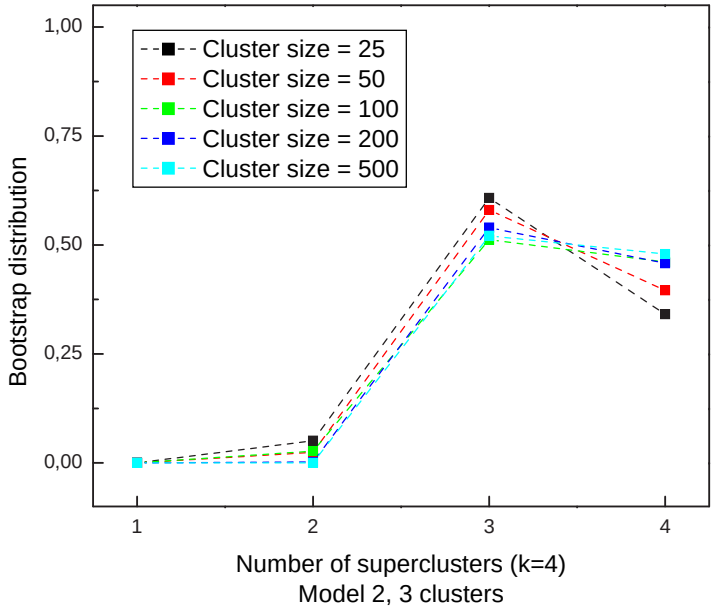


Figura 4.6: Modelo 2 con 3 clusters y diferentes tamaños de los clusters. Distribución bootstrap de p al aplicar k-medias con 4 clusters.

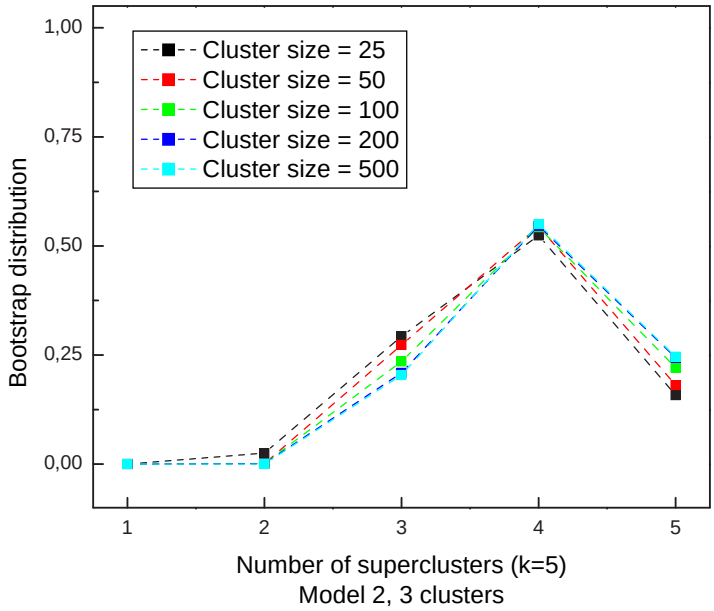


Figura 4.7: Modelo 2 con 3 clusters y diferentes tamaños de los clusters. Distribución bootstrap de p al aplicar k-medias con 5 clusters.

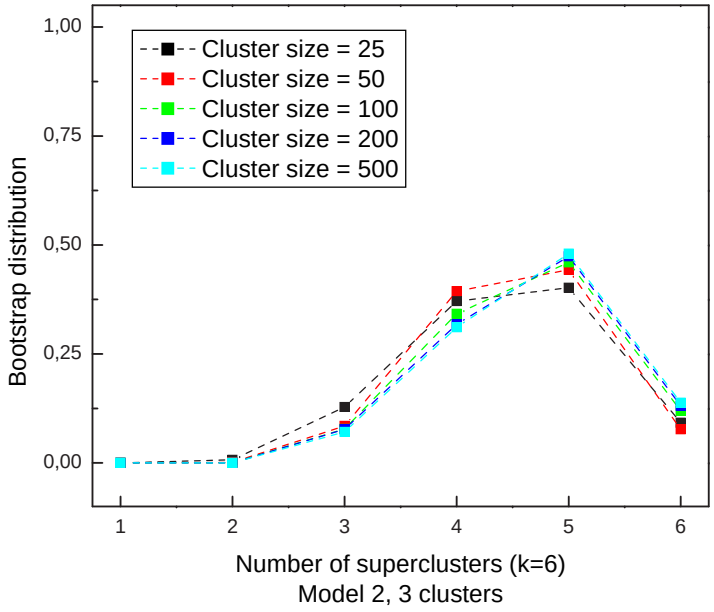


Figura 4.8: Modelo 2 con 3 clusters y diferentes tamaños de los clusters. Distribución bootstrap de p al aplicar k-medias con 6 clusters. ■

Estos resultados y los obtenidos en la sección 2.2.2, donde realizamos la comparación de clusterings con distinto número de clusters, muestran que no podemos esperar estimar el número de grupos de la población k con el número de súper-clusters obtenido tras comparar dos clusterings con un número cualquiera de clusters, pero sí son útiles para diseñar un heurístico para estimar k basándonos en múltiples comparaciones entre clusterings.

4.2.1. Muestreo en una población de clusterings

La idea básica es la siguiente: comenzando con una "población" de clusterings no jerárquicos obtenidos en un mismo conjunto de datos X, realizamos sucesivas comparaciones entre pares de dichos clusterings. La comparación de cada par conduce a un nuevo par de clusterings con un mismo número de clusters p. Estas comparaciones unen la información acerca de cómo se pueden agrupar los elementos de un conjunto, e intentan encontrar un consenso en una agrupación nueva. De esta manera, se espera que grupos reales que se dividieron incorrectamente en algunas de las particiones iniciales –por ejemplo, porque el número de clusters k utilizado en el algoritmo de clustering era mayor que el número real de grupos– se unan tras comparar el correspondiente clustering con otros en los que esos grupos no habían sido separados. Así, si genes que en algunas particiones pertenecen a diferentes clusters se agrupan juntos de manera persistente tras comparar diversos clusterings, asumiremos que efectivamente pertenecen a un mismo súper-cluster, y por otra parte, si genes que pertenecen a clusters distintos en algunas particiones siguen estando separados después de la mayor parte de las comparaciones, asumiremos que no comparten el mismo perfil y que por tanto deben permanecer en grupos diferentes.

Esta metodología tiene el inconveniente de ser totalmente dependiente de la calidad de los clusterings utilizados como población. En todas nuestras simulaciones hemos empleado k-

medias; como explicamos en la introducción, existen otros métodos, como el algoritmo PAM Kaufman y Rousseeuw (1990), que han demostrados ser más robustos y eficientes que k-medias. Además, este método tiene la ventaja adicional de proporcionar un gráfico, la representación del estadístico silueta, que puede utilizarse para seleccionar el número de clusters óptimo. La razón por la que a pesar de ello utilizamos k-medias en nuestra intención de estimar el número de grupos es que, siguiendo la línea de refinamiento del capítulo 3, pretendemos mejorar, al mismo tiempo, la falta de robustez de los resultados obtenidos con k-medias usando las sucesivas comparaciones.

De manera más precisa, consideremos que la partición real de los datos en k grupos ($k \geq 1$) está descrita por C_{real} y sea C una partición obtenida por algún procedimiento de clustering. Si representamos ambas colecciones de clusters como nodos en un bigrafo, tal y como hicimos en el capítulo 2, y aplicamos el algoritmo greedy, obtendremos p componentes conexas. Si $p = k$, diremos que C es una "buena" partición de X; en caso contrario, (i.e. $p < k$) diremos que C es una "mala partición", puesto que no es capaz de separar algunos de los grupos reales (véase la figura 4.9.a).

Cuando dos particiones C_1 y C_2, con m y n clusters respectivamente, se comparan mediante el algoritmo greedy, las dos particiones resultantes tendrán $p \leq \text{mín}\{m, n\}$ clusters. Si tanto C_1 como C_2 son "buenas" particiones, lo más común es que las nuevas particiones también sean "buenas"; en cambio, si alguna de las particiones une dos grupos reales en un único cluster, tras la comparación, las dos nuevas particiones también lo harán (ver las figuras 4.9.b y c).

Existen casos en los que la comparación de un clustring "malo" y otro "bueno" no hace que el segundo se contamine. Por ejemplo, consideremos una partición real C_{real} con dos grupos, y un par de clusterings, uno "bueno" y otro "malo". Supongamos que el "mal" clustering **A**

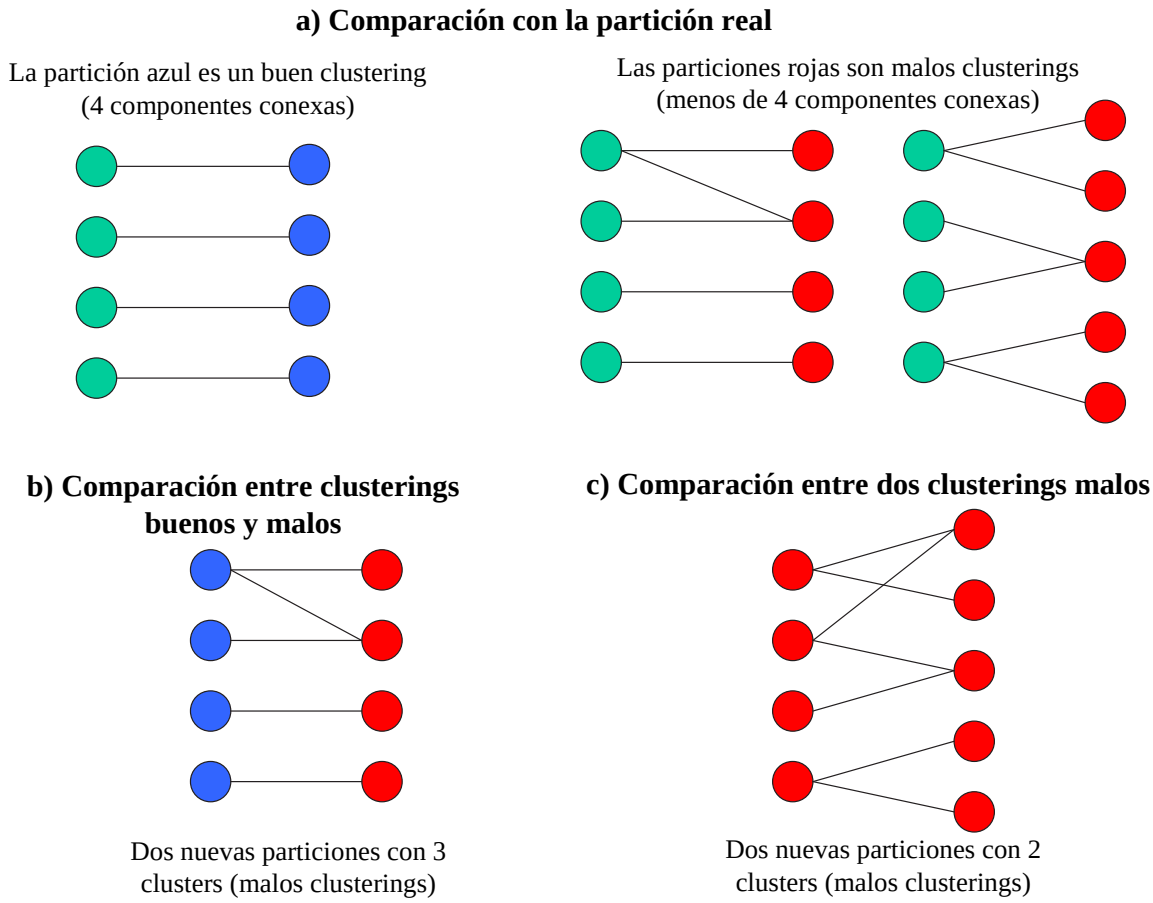


Figura 4.9: Clusterings "buenos" y "malos". a) Los nodos verdes representan la partición real C_{real} de los datos; los nodos azules corresponden a un "buen" clustering mientras que los nodos rojos corresponden a dos "malos" clusterings. b) La comparación de un "buen" clustering y un "mal" clustering conduce a dos "malos" clusterings. c) La comparación de dos "malos" clusterings conduce a dos "malos" clusterings.

tiene tres clusters $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$, uno de los cuales consiste en un gen g_1 procedente de uno de los grupos reales, y otro gen g_2 procedente del otro grupo, y los otros dos clusters contienen respectivamente genes de sólo uno de los grupos reales. Este pequeño cluster de dos elementos hace que el clustering sea "malo", pues al compararlo con la partición real, obtendremos una

única componente en el grafo. Consideremos también el clustering "bueno" **B** con dos grupos B_1, B_2, en los que sólo el gen g_1 ha sido erróneamente asignado (véase la figura 4.10). La comparación de ambos clusterings conduce a un bigrafo con dos componentes conexas, que se corresponden con los dos grupos reales, y los dos nuevos clusterings coincidirán con **B**; por tanto, la comparación entre un clustering "bueno" y otro "malo", en este caso produce dos clusterings "buenos".

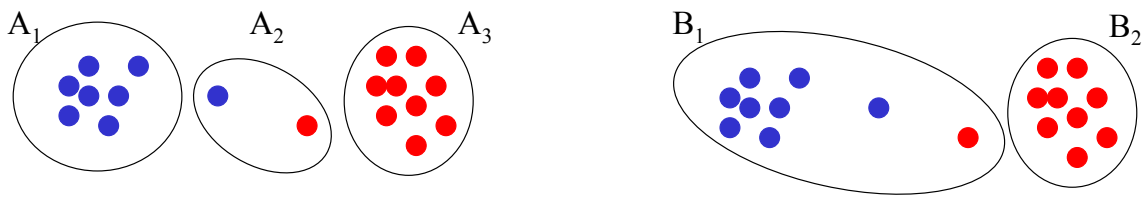


Figura 4.10: El clustering **A**, a la izquierda, es un "mal" clustering puesto que al ser comparado con la partición real produce una única componente conexa en el grafo que los representa. El clustering **B**, a la derecha, es un "buen" clustering.

Sin embargo, la proporción de veces que en la práctica ocurre este tipo de conversión es tan pequeña que asumiremos que la comparación entre dos de tales particiones "buenas" conduce a otras dos particiones "buenas", y que la comparación con al menos un clustering "malo" conduce a dos clusterings "malos" con probabilidad 1.

Bajo esta suposición, se verifica la siguiente proposición:

Proposición 4.1 *Si en una población de $r \geq 2$ particiones existen x particiones "buenas", la comparación de todas frente a todas produce una proporción de clusterings "malos" mayor que la obtenida al muestrear con remplazamiento l pares de clusterings de la población, con l suficientemente grande.*

Demostración: Si comparamos todas con todas, obtendremos $x(x-1)$ particiones "buenas" entre los $r(r-1)$ nuevos clusterings. (Obsérvese que el número de particiones "malas" puede ser dramáticamente grande, incluso si $x \geq r - x$).

Por otro lado, si denotamos por X_i la variable aleatoria que computa el número de particiones "buenas" después de comparar un par aleatorio,

$$X_i = \begin{cases} 2, & \text{con probabilidad } \dfrac{x^2}{r^2} \\ 0, & \text{con probabilidad } 1 - \dfrac{x^2}{r^2} \end{cases}, \tag{4.2.1}$$

el número esperado de particiones "buenas" en l pares será:

$$G = E\left[\sum_{i=1}^{l} X_i\right] = 2l\frac{x^2}{r^2}.$$

Por tanto, la proporción esperada de particiones "buenas" después de l comparaciones será:

$$\frac{\dfrac{2lx^2}{r^2}}{2l} = \frac{x^2}{r^2} \tag{4.2.2}$$

que no depende de l y, de forma trivial se comprueba que es mayor que la proporción resultante tras comparar cada par posible, sin remplazamiento, $\dfrac{x(x-1)}{r(r-1)}$, suponiendo que el número de particiones "buenas" es estrictamente positivo. ■

Así, el algoritmo que proponemos tiene como objetivo estimar el verdadero número de grupos aplicando k-medias muchas veces, con distinto número de clusters, k_{init_i}, y muestreando (con remplazamiento) esta muestra de clusterings. Después, pares aleatorios de estas particiones de los datos se comparan y re-etiquetan según el número de súper-clusters resultante de aplicar el algoritmo greedy.

Existe trabajo previo basado en la replicación de la asignación a clusters (ver e.g. Breckenridge (1989) y Dudoit y Fridlyand (2002)), pero en tales estudios, el conjunto remuestreado es el de los datos. En cambio, nosotros construimos un conjunto de clusterings y obtenemos muestras de este conjunto. Obsérvese que los objetivos de ambos métodos son diferentes: nosotros no intentamos asignar cada gen a un cluster en particular, sino que simplemente usamos la comparación de clusterings para estimar cuántos de tales clusters existen.

4.2.2. Método de muestreo de clusterings

Consideremos el conjunto X con N observaciones (genes), y seleccionemos un algoritmo de clustering no determinista (inicializado con algún parámetro aleatorio) que divida a X en k_{init} clusters (e.g. 'k-medias'). Cada iteración b del algoritmo de muestreo procede en tres pasos. Se fijan de antemano unos valores enteros máximo y mínimo del número de clusters usado, k_{max} y k_{min}. En lo que sigue, usaremos k-medias como el método de clustering.

Cuadro 4.2.2.1. Los tres pasos del algoritmo *sampling_estimation*().

algoritmo *sampling_estimation*(X, $k_{min}, k_{max}, n_1, n_2, n_3, B$);

1. FOR $b = 1 : B$

PASO 1.

S1.1. Aplicar k-medias $2 \times n_1$ veces, con k_{init} clusters elegidos aleatoriamente en $[k_{min}, k_{max}]$.

S1.2. Crear n_1 pares de clusterings, $\{C_{1,f_i}, C_{2,f_i}\}$, $(i = 1,\ldots, n_1)$.

S1.3. Para cada par $\{C_{1,f_i}, C_{2,f_i}\}$, aplicar el algoritmo greedy, determinar el correspondiente número de súper-clusters p_{f_i} y unir los clusters iniciales para obtener las dos nuevas particiones, $\{C^*_{1,f_i}, C^*_{2,f_i}\}$, con el mismo número de clusters.

S1.4. Elegir aleatoriamente uno de estos dos nuevos clusterings como parte de la muestra de clusterings, S_1, de tamaño n_1, usada en el paso 2.

PASO 2.

S2.1. Remuestrear n_2 pares de clusterings con remplazamiento en S_1.

S2.2. Comparar estos pares usando el algoritmo greedy y obtener nuevos pares de clusterings $\{C^*_{1,s_i}, C^*_{2,s_i}\}, i = 1, ..., n_2$, con el mismo número de clusters.

S2.3. Para cada par, seleccionar aleatoriamente uno de los elementos para formar parte de la nueva muestra S_2.

S2.4. Determinar el número de elementos más frecuente de súper-clusters en S_2, p_{freq}, y encontrar el subconjunto $S^*_2 \subset S_2$ que contiene sólo los clusterings con p_{freq} clusters.

PASO 3.

S3.1. Remuestrear n_3 pares en S^*_2 para conformar la muestra final S_3.

S3.2. Utilizar el algoritmo greedy para obtener los nuevos pares $\{C^*_{1,t_i}, C^*_{2,t_i}\}$, $i = 1, ..., n_3$.

S3.3. Determinar el número más frecuente de súper-clusters en la nueva muestra S_3 para estimar el verdadero número de grupos en la b-ésima iteración, $\hat{k_b}$.

2. Estimar k como $\hat{k} = \text{Moda}_{b=1,\ldots,B}\left(\hat{k_b}\right)$.

El algoritmo *sampling_estimation*() se describe formalmente en el cuadro 4.2.2.1. Los parámetros n_1, n_2, n_3, para los tamaños muestrales, y B, para el número máximo de iteraciones, son también enteros especificados de antemano.

Cada paso del algoritmo de muestreo desempeña un papel diferente. El primer paso crea una población inicial heterogénea de clusterings con información acerca de la estructura de los datos. Esta información no siempre es precisa debido al comportamiento de k-medias. El segundo paso tiene como objetivo encontrar similaridades entre los diferentes clusterings y selecciona el subconjunto de clusterings que conduce al número de súper-clusters más común. El tercer paso, cuya necesidad se basa en lo observado en las secciones 2.2.2 y 4.2, está diseñado para confirmar la estabilidad de los resultados obtenidos en el segundo paso, o bien para reagrupar clusters si la muestra S_2^* no es suficientemente homogénea. De hecho, cuando el número de clusters m y n en las particiones es mayor que el verdadero k, el cálculo de los super-clusters hace que estos números estén más próximos al verdadero k, y cuando m y n están muy próximos a k el número más probable de súper-clusters es k, como se observó en las distribuciones bootstrap de p.

Dependiendo de los valores que seleccionemos para k_{min} y k_{max} podemos definir dos enfoques diferentes.

a) Aplicar sucesivamente el algoritmo para un rango de valores de k_{init}, haciendo $k_{init} = k_{min} = k_{max}$. Así, podemos obtener un gráfico del número de clusters iniciales k_{init} frente a la correspondiente estimación $\hat{k}_{init}$ y seleccionar la mejor estimación k_{single} según un determinado criterio (e.g., el valor más frecuente, como se mencionó anteriormente, o un valor en el que el gráfico se estabiliza). A este

método lo denominamos k-único (single k).

b) Aplicar el algoritmo una sola vez, seleccionando k_{init} en un conjunto de valores apropiado, con $k_{min} < k_{max}$. En este caso, la estimación la denotamos por k_{mixed}, y el método lo denominamos k-mixto (mixed k).

El método *k-único* tiene la ventaja de proporcionar una visualización gráfica y de calcular un estimador para diversos valores iniciales, lo cual puede aumentar nuestra confianza en el resultado, mientras que el método *k-mixto* es más rápido que el primero. En cualquier caso, la principal desventaja de ambos enfoques es que el método requiere valores de k_{init} mayores que el número real de grupos, que es desconocido (ésta es la única restricción para que el algoritmo greedy sea eficiente); en caso contrario, todos los clusterings obtenidos son "malos", pues todos ellos unirán grupos reales. Una manera de evitar este problema consiste en seleccionar un rango de valores lo suficientemente amplio para cubrir el número real de grupos. A menudo, esto no es una gran restricción ya que el conocimiento (biológico) previo acerca de los datos puede dar una idea aproximada de lo grande que es este número. Otra alternativa es combinar ambos métodos: usar el *k-único* para encontrar un rango de valores que analizar mediante el método *k-mixto*, como se verá en la sección 4.4.1.

4.3. Método del diámetro de los puntos más profundos

Como se apuntó anteriormente, el algoritmo que proponemos a continuación se basa en elegir una proporción prefijada de puntos en cada cluster (siguiendo un criterio de profundidad) y en minimizar alguna medida del tamaño espacial de los subconjuntos así obtenidos; en particular,

hemos elegido como tal medida la suma de los diámetros de los subconjuntos. La idea de minimizar la suma de diámetros no es nueva en el análisis cluster. Como ejemplo, véanse los trabajos de Charikar y Panigrahy (2004); Doddi *et al.* (2000) y Monma y Suri (1991).

La novedad que introducimos para estimar el número de clusters es muy intuitiva. Es muy frecuente tener datos atípicos en el conjunto de datos, lo cual puede hacer que el diámetro de éste sea dramáticamente grande. Al tomar sólo un porcentaje dado de los puntos más profundos se obtienen conjuntos más pequeños y compactos, pues esos datos atípicos no son tenidos en cuenta. En el caso del análisis cluster, si tenemos menos clusters que el número real k, la situación más común es que estemos uniendo al menos dos de ellos. Entonces, es natural que los datos más profundos en uno de estos clusters no sean representativos de un único cluster, pues pueden pertenecer a regiones alejadas de los respectivos "centros", y formarán un subconjunto disperso. Por tanto, la suma de los diámetros será mayor que en el caso de aplicar el algoritmo de clustering con k clusters. Por otra parte, si el número de clusters es mayor que el verdadero k, entonces, incluso si los puntos más profundos forman conjuntos compactos, la suma de los diámetros aumentará ligera pero progresivamente.

Existen diversas maneras de calcular el diámetro de un conjunto X. Consideremos por ejemplo:

- el diámetro completo, que calcula la distancia entre los puntos más alejados contenidos en X:

$$\mathbb{D}_{complete}(\mathrm{X}) = max_{x,y\in \mathrm{X}} d(x,y)$$

- el diámetro medio, que calcula la distancia media entre todos los puntos de X:

$$\mathbb{D}_{average}(\mathrm{X}) = \frac{\sum_{x,y\in \mathrm{X}, x\neq y} d(x,y)}{\binom{|\mathrm{X}|}{2}}$$

- el diámetro por centroides, que calcula el doble de la distancia media entre todos los puntos y el centroide μ_{X} de X:

$$\mathbb{D}_{centroid}(\mathrm{X}) = \frac{2}{|\mathrm{X}|} \sum_{x \in \mathrm{X}} d(x, \mu_{\mathrm{X}})$$

Con las tres medidas obtuvimos resultados similares; en este capítulo sólo mostraremos los correspondientes al diámetro completo.

El algoritmo que proponemos, al que nos referiremos como *diámetro recortado* (trimmed diameter), se puede describir con pseudo-código como sigue. Los datos que se requieren como entrada son el conjunto X, el número máximo de clusters K y el porcentaje α de puntos más profundos considerado.

Cuadro 4.3.2. El algoritmo "diámetro recortado", *trimmed_diameter*(), para estimar el número de clusters.

algoritmo *trimmed_diameter*(X, K, α);

1. FOR k in 1:K

   Aplicar algún algoritmo de clustering a X para obtener k clusters.

   En cada cluster, encontrar el $\alpha\,\%$ de puntos más profundos y formar con éstos los subconjuntos $Y_1, ..., Y_k$.

   Calcular la suma de diámetros $\mathbb{D}_k = \sum_{i=1}^{k} \mathbb{D}(Y_i)$.

2. Estimar el número de clusters con el $\hat{k}$ que minimiza la suma de diámetros:

$$\hat{k} = argmin_{k=1,...,K} \mathbb{D}_k.$$

Este simple algoritmo conduce a buenos resultados en una gran variedad de conjuntos, como veremos en la sección 4.4, en la que aplicamos los algoritmos descritos anteriormente tanto a datos simulados como a datos reales.

4.4. Resultados

4.4.1. Datos simulados

En nuestro estudio de simulación, utilizamos algunos de los modelos usados en Tibshirani *et al.* (2001); Dudoit y Fridlyand (2002) y en el capítulo 2. Para cada modelo simulado, generamos 50 conjuntos y les aplicamos los siguientes métodos para estimar el número de grupos: el estadístico silueta, el estadístico gap, el estadístico gapPC, el método CLEST, el estadístico ReD, el método k-único, el método k-mixto y el algoritmo del diámetro recortado. El criterio que seguimos para estimar el número de grupos con el método k-único es seleccionar la estimación más frecuente en el gráfico k_{init} vs $\hat{k}$, deshaciendo empates con el $\hat{k}$ más pequeño. Cuando utilizamos el método k-mixto, la decisión más importante que hay que tomar es la selección del rango en el cual escogeremos k_{init} para k-medias. Como dijimos anteriormente, un requisito para conseguir una elección apropiada es que el intervalo $I = [k_{min}, k_{max}]$ incluya valores mayores que el verdadero k; en principio, podríamos seleccionar k_{max} arbitrariamente grande, pero valores excesivamente altos para k_{max} pueden derivar en una sobre-estimación de k. Por otra parte, k_{min} no debería ser mucho mas pequeño que k, pues los "malos" clusterings contaminan los "buenos" y la proporción de éstos decrece rápidamente, lo cual conduce a malas estimaciones. Obsérvese que esto se puede evitar en el método k-único simplemente descartando la estimación producida por estos valores iniciales, pues k_{min} y k_{max} pueden ser reajustados en cualquier momento del algoritmo. En cambio, cualquier cambio necesario en estos valores

en el algoritmo k-mixto implica una nueva ejecución del algoritmo.

Por otra parte, una mala selección de k_{min}, –en el sentido de incluir muchos "malos" clusterings, i.e., con k_{min} muy pequeño–, no se puede solventar aumentando el tamaño de las muestras n_1, n_2 y n_3, ya que la proporción de "buenos" clusterings (4.2.2) no depende de estos valores. Por tanto, no se puede conseguir una estimación mejor en general. De hecho, la heterogeneidad de las muestras finales S_3 obtenidas usando diferentes valores de n_1, n_2 y n_3 es muy similar, así como el porcentaje de elementos mal asignados, y por tanto, también lo serán las correspondientes estimaciones.

Bajo la hipótesis de que tenemos alguna información biológica sobre los datos particulares que estamos estudiando y que podemos tener una idea aproximada del número de grupos que existen, debemos seleccionar k_{min} próximo a (y/o mayor que) este valor para obtener una estimación precisa. Si no tenemos dicha información, podemos aplicar el método k-único para encontrar un intervalo apropiado. En particular, si al aplicar el k-único obtenemos la estimación $\hat{k}_{single}$, haremos:

$$k_{min} = \hat{k}_{single} \tag{4.4.3}$$

$$k_{max} = \hat{k}_{single} + \kappa, \tag{4.4.4}$$

donde κ es un entero positivo. En nuestros ejemplos, utilizamos $\kappa = 2 \times \hat{k}_{single}$, para obtener $k_{max} = 3 \times \hat{k}_{single}$.

Para cada conjunto de datos, aplicamos el algoritmo utilizando distintos intervalos $I = [k_{min}, k_{max}]$, para ilustrar el comportamiento debido a selecciones apropiadas o inapropiadas. En particular, si forzamos a k_{min} y k_{max} a tomar valores que aumentan la proporción de "malos" clusterings y que segregan en exceso los grupos reales, haciendo difícil la recuperación

de éstos, respectivamente, los intervalos resultan inapropiados.

Finalmente, para el método del diámetro recortado, seleccionamos $\alpha = 50\,\%$.

Como medida de la precisión de cada método, calculamos el porcentaje de simulaciones en las que se encontró el verdadero número de grupos.

Ejemplo 4.3 *Modelo 1 de tamaño 100*

Para el modelo 1, seleccionamos conjuntos con 4 grupos reales de 100 elementos cada uno y distintos niveles de ruido blanco $\sigma = 1, ..., 6$. Para cada conjunto, aplicamos el algoritmo k-único, empezando con $k_{init} = 2, ..., 12$ clusters en k-medias, con tamaños muestrales $n_1 = 10$, $n_2 = 100$ y $n_3 = 100$, y con el parámetro de remuestreo $B = 100$.

Para ilustrar el comportamiento del método realizamos un gráfico con los resultados de un conjunto en particular, para cada valor de σ, en la figura 4.11.a). También obtuvimos el porcentaje de elementos (genes) mal asignados comparando los clusters resultantes de cada elemento de la muestra S_3 con los grupos reales generados, usando el criterio del voto por mayoría; estos porcentajes se muestran en la figura 4.11.b). Claramente, el error es grande cuando la estimación está por debajo del verdadero k, pero cuando la estimación alcanza este valor, el error cae rápidamente, mostrando que la estimación de k es correcta y precisa, y que los súper-clusters encontrados están próximos a los grupos reales.

La figura 4.11.c) muestra los valores medios obtenidos para el índice de Rand cuando comparamos cada clustering de la muestra S_3 con la partición real que habíamos generado.

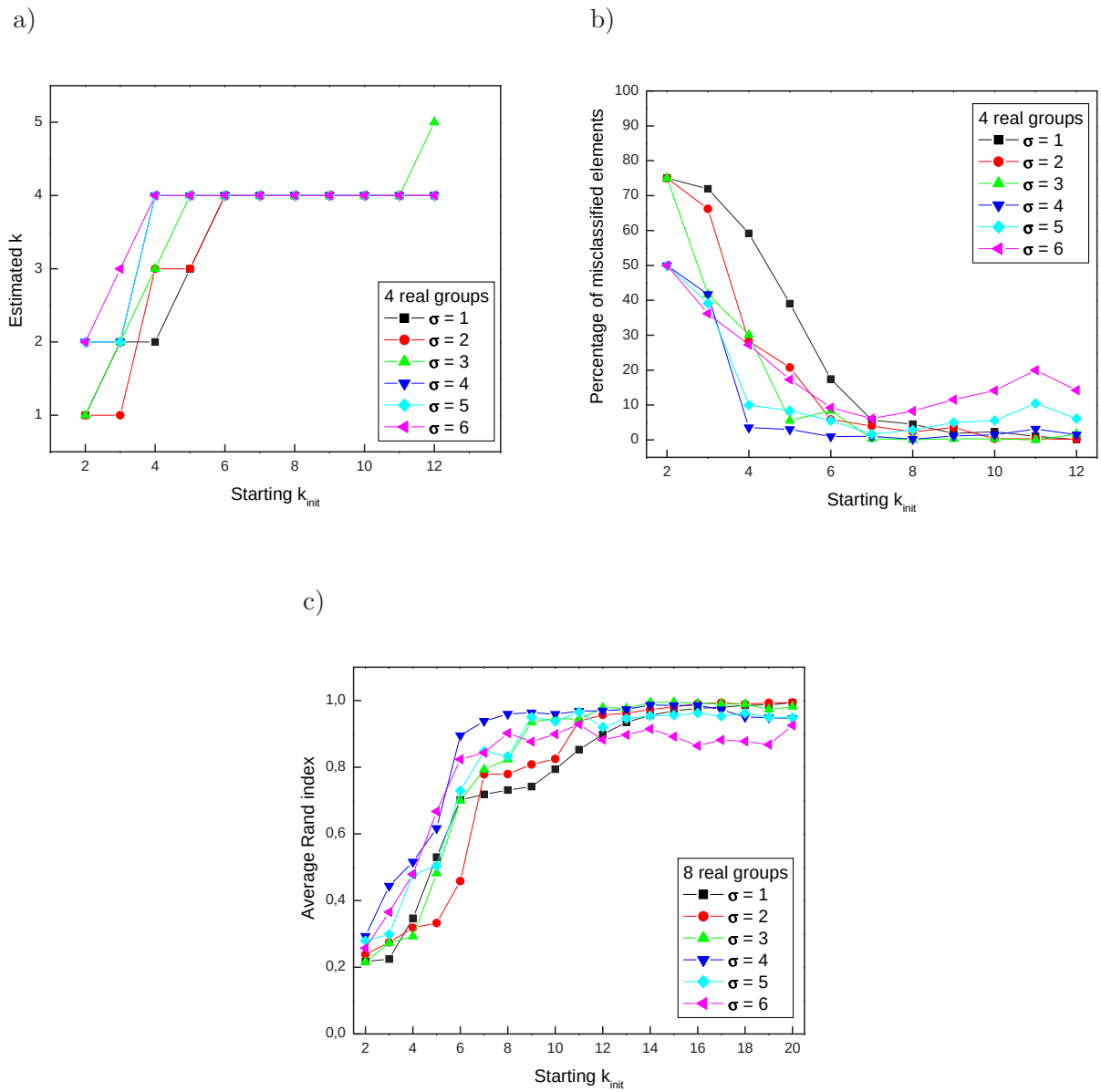
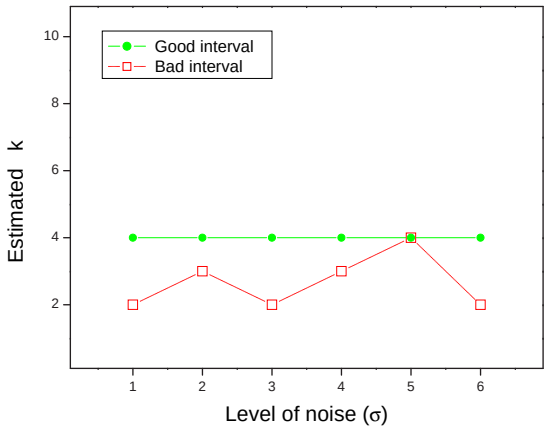
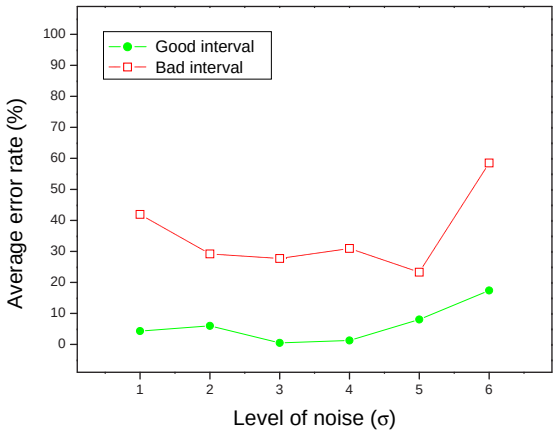


Figura 4.11: a) Estimación del número real de grupos, con diferentes niveles de ruido y diferentes valores de k_{init} en k-medias, con el método k-único. b) Porcentaje de elementos mal clasificados después de obtener los súper-clusters, según el criterio del voto por mayoría, para los mismos parámetros. c) Valores medios del índice de Rand para los distintos niveles de ruido, obtenidos al comparar los clusterings de la muestra S_3 y la verdadera partición.

a)



b)



c)

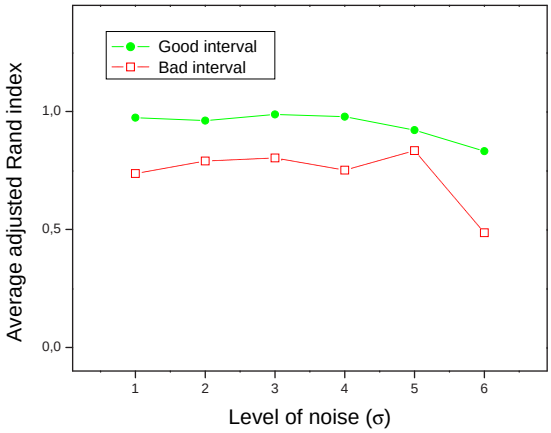


Figura 4.12: a) Estimación del número real de grupos, con diferentes niveles de ruido e intervalos $I = (k_{min}, k_{max})$, con el método k-mixto. Los círculos sólidos y las líneas continuas corresponden a la estimación obtenida usando el intervalo correcto (4,12). Los círculos abiertos y las líneas discontinuas representan la estimación utilizando el intervalo incorrecto (2,10). b) Porcentaje correspondiente de elementos mal clasificados, según el criterio del voto por mayoría. c) Índice de Rand calculado para cada conjunto.

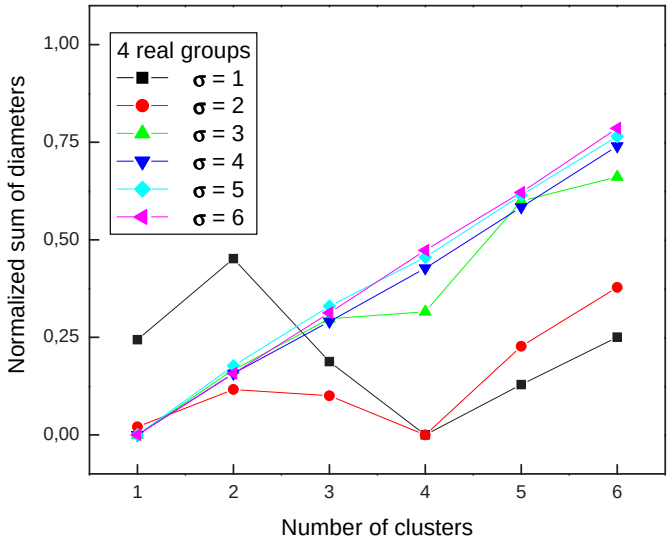


Figura 4.13: Suma normalizada de los diámetros para el 50 % de los puntos más profundos de cada cluster para datos simulados según el modelo 1, con 4 grupos y diferentes niveles de ruido, al variar el número de clusters desde 1 hasta 6.

De forma similar al caso de los porcentajes de mala asignación, la concordancia entre los clusterings producidos por k-medias y la verdadera partición es muy buena cuando se alcanza la estimación correcta, mientras que en el caso de subestimar k, la similaridad entre éstos es muy pobre.Obsérvese que en los casos en los que k está sobre-estimado, la concordancia también es buena, pues algunos grupos son divididos, y no unidos de forma errónea.

Usamos el mismo ejemplo para ilustrar el comportamiento del método k-mixto. En este caso, utilizamos diferentes valores de k_{min} y k_{max} para obtener intervalos apropiados e inapropiados. En particular, para los primeros elegimos $I = (4, 12)$, siguiendo las reglas 4.4.3 y 4.4.4, ya que la estimación dada por el k-único en estos conjuntos de datos era 4, y para los segundos $I = (2, 10)$. Los tamaños muestrales y el número de remuestreos son de nuevo $n_1 = 10$, $n_2 = 100$, $n_3 = 100$ y $B = 100$. En el primer caso, independientemente del nivel de ruido, la estimación siempre es correcta, mientras que en el segundo caso, ésta oscila entre 2 y 4, como se muestra en la figura 4.12. Para un número mayor de grupos, el comportamiento del método con intervalos inapropiados es mucho peor, oscilando entre 1 y 2 para conjuntos con 6 grupos y entre 1 y 3 para conjuntos con 8 grupos (resultados no mostrados).

Finalmente, la figura 4.13 muestra los valores, normalizados entre 0 y 1, de la suma de los diámetros cuando el número de clusters k oscila entre 1 y 6 en el algoritmo del diámetro recortado. En este caso, podemos ver que el algoritmo es capaz de encontrar los verdaderos grupos sólo para niveles bajos de ruido, en los que no existe un gran solapamiento entre los clusters cuando éstos son representados en coordenadas paralelas.

A continuación, y como ilustración, mostramos en la tabla 4.1 el resumen del número de clusters óptimo estimado por cada uno de los métodos, cuando $\sigma = 1$ y 3. El número real de grupos está señalado con *. Los mejores métodos son el estadístico gap (GAP), el estadístico gapPC (GAPPC), el método CLEST (CLEST), el algoritmo k-único (SK) y el k-mixto (MK). El diámetro recortado (TD) funciona bien para bajos niveles de ruido, pero a medida que dicho nivel aumenta comienza a comportarse de manera pobre, como ya se señaló en la figura 4.13. Lo mismo ocurre con el estadístico silueta (SIL), pero para $\sigma = 3$ mejora a TD. ReD falla en todos los casos, detectando un único cluster.

a)

Método	1	2	3	4*	5	6
SIL	0	0	3	47	0	0
GAP	0	0	0	**50**	0	0
GAPPC	4	0	0	46	0	0
CLEST	0	0	0	**50**	0	0
RED	50	0	0	0	0	0
SK	0	0	0	**50**	0	0
MK	0	0	0	**50**	0	0
TD	0	0	1	49	0	0

b)

Método	1	2	3	4*	5	6
SIL	0	4	15	31	0	0
GAP	0	0	0	**50**	0	0
GAPPC	0	0	0	**50**	0	0
CLEST	0	0	0	**50**	0	0
RED	50	0	0	0	0	0
SK	0	0	0	**50**	0	0
MK	0	0	0	**50**	0	0
TD	35	1	0	14	0	0

Tabla 4.1: a) Número estimado de clusters para cada método en el modelo 1, con 4 grupos, $\sigma = 1$ y tamaños de los clusters de 100. b) Número estimado de clusters para cada método en el modelo 1, con 4 grupos, $\sigma = 3$ y tamaños de los clusters de 100. ■

Ejemplo 4.4 *Modelo 2*

Estos conjuntos de datos tienen 3 clusters en dimensión 2; los clusters corresponden a variables normales bivariantes, independientes, con 25, 25 y 50 observaciones, centradas en $(0, 0)$, $(0, 5)$ y $(5, -3)$, y con matriz de varianzas covarianzas igual a la matriz identidad.

Para el método SK usamos $k_{min} = 2$, $k_{max} = 8$, y también para el método MK utilizamos

$n_1 = 10$, $n_2 = 100$, $n_3 = 100$ y $B = 100$. La tabla 4.2 muestra la estimación obtenida con cada método. Todos ellos funcionan bien excepto ReD, que sólo encuentra un grupo en todos los conjuntos de datos generados.

Método	1	2	3*	4	5	6
SIL	0	0	**50**	0	0	0
GAP	0	0	**50**	0	0	0
GAPPC	0	0	**50**	0	0	0
CLEST	0	2	48	0	0	0
RED	50	0	0	0	0	0
SK	0	0	42	8	0	0
MK	0	0	**50**	0	0	0
TD	0	0	**50**	0	0	0

Tabla 4.2: Número estimado de clusters para cada método en el modelo 2 (3 grupos).

■

Ejemplo 4.5 *Modelo 3*

Estos conjuntos de datos están formados por un único cluster en dimensión 10, con 200 observaciones uniformemente distribuidas en el hipercubo unidad de dimensión 10. Los resultados correspondientes se muestran en la tabla 4.2. Obsérvese que SIL no es capaz de encontrar

menos de 2 grupos, por lo que es el peor de los métodos utilizados, seguido por SK y MK, el cual, en cualquier caso, consigue mejorar considerablemente los resultados de SK. Los mejores algoritmos para este modelo son RED, CLEST y TD, los cuales estiman el verdadero valor en todos los conjuntos simulados. GAP y GAPPC también se comportan muy bien, fallando sólo en 6 y 2 conjuntos, respectivamente.

Método	1*	2	3	4	5	6
SIL	0	29	4	5	6	6
GAP	44	6	0	0	0	0
GAPPC	48	2	0	0	0	0
CLEST	**50**	0	0	0	0	0
RED	**50**	0	0	0	0	0
SK	10	36	4	0	0	0
MK	38	10	2	0	0	0
TD	**50**	0	0	0	0	0

Tabla 4.3: Número estimado de clusters para cada método en el modelo 3 (1 grupo).

■

Ejemplo 4.6 *Modelo 4*

Los datos contienen 4 clusters en dimensión 10, con 7 variables ruido. Para cada cluster se escoge aleatoriamente el número de observaciones entre 25 o 50; las observaciones en un cluster determinado tienen distribución normal multivariante con matriz de varianzas-covarianzas la

identidad, y son independientes. Para cada cluster, la media de las primeras tres variables se escogen de una distribución $N(0_3, 25I_3)$, donde 0_p denota un vector $1 \times p$ de ceros e I_p denota la matriz identidad $p \times p$. Las medias en las restantes siete variables son 0. Se descartó cualquier simulación en la que la distancia euclídea entre las dos observaciones más próximas pertenecientes a dos clusters diferentes fuese menor que 1.

Método	1	2	3	4*	5	6
SIL	0	0	4	46	0	0
GAP	4	0	0	46	0	0
GAPPC	8	0	0	42	0	0
CLEST	0	1	15	34	0	0
RED	0	31	18	1	0	0
SK	0	15	10	25	0	0
MK	0	0	1	49	0	0
TD	0	0	0	**50**	0	0

Tabla 4.4: Número estimado del número de clusters para cada método en el modelo 4 (4 grupos).

Después de aplicar todos los métodos, obtuvimos la tabla 4.4. Como se puede ver, el mejor método es TD, seguido por MK, SIL y GAP. El peor comportamiento corresponde a RED, CLEST y SK, siendo el primero de éstos mucho peor que los demás. ■

Como reflejan estos resultados, no existe ningún método entre los utilizados que se comporte mejor que los demás en todos los ejemplos propuestos.

La ventaja de utilizar el método k-único es que los límites k_{min} y k_{max} se pueden reajustar dependiendo de los valores obtenidos, descartando los valores iniciales o bien utilizando más pasos en el algoritmo. En general, hasta unas tres veces el número real de grupos, el método proporciona buenas estimaciones. Una desventaja del método es que puede resultar difícil escoger una buena estimación, ya que hay situaciones en las el gráfico no se estabiliza, o en las que no existe una elección claramente mejor, pues deshacer empates con el menor valor, obviamente puede no ser el mejor criterio. La combinación de la información contenida en clusterings con distinto número de clusters en el método k-mixto proporciona un estimador, con una única iteración del algoritmo de muestreo, y que a menudo es muy precisa.

Por otra parte, el uso del método del diámetro recortado lleva a resultados precisos de forma rápida (porque la noción de profundidad utilizada no es computacionalmente costosa) en la mayor parte de los casos. No obstante, observamos que falla cuando los datos, considerados como funciones discretas, no son suaves y al representarlas en coordenadas paralelas los grupos tienen un alto grado de superposición.

En la sección siguiente aplicaremos estos métodos a datos reales.

4.4.2. Datos reales

Para comprobar cómo se comportan los tres métodos propuestos en datos reales, utilizamos el siguiente conjunto de datos:

Ejemplo 4.7 *Datos de la salida de la fase estacionaria en Schizosaccharomyces cerevisiae*

Los datos de la salida de la fase estacionaria en *S. cerevisiae* (Radonjic *et al.*, 2005) contiene 6357 genes cuyos perfiles de expresión fueron medidos en 34 muestras, utilizando experimentos

de microarrays, para descubrir múltiples transiciones durante un ciclo completo de crecimiento de 9 días entre la salida y la entrada en la fase estacionaria. Este ciclo comprende una fase de demora, una fase exponencial, un cambio diauxico, una fase post-diauxica y una fase estacionaria (ver la figura 4.14).

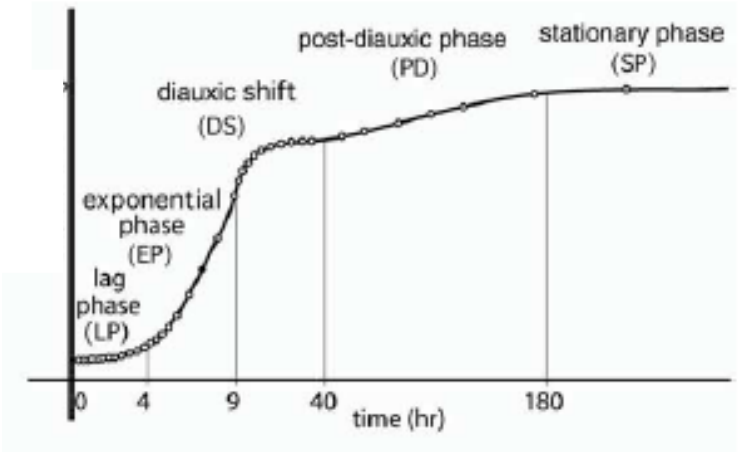


Figura 4.14: Representación esquemática de la salida y entrada en la fase estacionaria en S.cerevisiae, incluyendo las etapas intermedias, en un único experimento de un cultivo que abarca 9 días.

Cada gen estaba representado dos veces. Los datos se procesaron como sigue: en primer lugar, se normalizaron utilizando lowess y a continuación todos los valores por debajo de 50 se truncaron a este valor. Posteriormente, se utilizó un spline cúbico (implementado en R) para suavizar los datos y por último, se ponderaron los valores de los genes duplicados y se tomó el logaritmo de los valores resultantes. Mediante Expression Profiler NG (Kapushesky *et al.*, 2004), seleccionamos los genes que presentaban al menos un 5 % de las condiciones separadas de la correspondiente media más de dos veces la desviación típica, lo cual llevó a un conjunto de 520 genes en 34 condiciones.

Aplicamos todos los métodos anteriores a este conjunto reducido. Para el método SK utilizamos $k_{init} = 2, ..., 10$ clusters con k-medias, obteniendo una estimación de 5. Aplicamos k-medias con k=5 clusters para intentar encontrar alguna información biológica. Los cinco clusters que obtuvimos están representados en la figura 4.15, junto a los perfiles (medias) de los clusters. Como se puede ver, son bastante coherentes con los resultados publicados en Radonjic *et al.* (2005). Por ejemplo, el cluster 1, en negro, corresponde a genes inducidos al final de la fase de demora; el cluster 2, en rojo, está enriquecido con genes con actividad en la fase estacionaria, además de incluir genes regulados positivamente al final del cambio diauxico, pero que aún permanecen activos en la fase estacionaria, mientras que el cluster 3, en verde, incluye genes que fluctúan varias veces en el tiempo.

Utilizando la estimación de SK, aplicamos el método MK según las reglas 4.4.3 y 4.4.4, y determinamos $k_{min} = 5$ y $k_{max} = 15$. De nuevo, obtuvimos una estimación de 5.

Los otros métodos fueron aplicados permitiendo un número máximo de clusters de 8. El estadístico silueta proporcionó un valor de 2, claramente subestimando el número de grupos en el conjunto, como muestra la investigación realizada en Radonjic *et al.* (2005). También RED y TD subestiman el número de clusters con $\hat{k} = 1$. CLEST coincide con el resultado obtenido con SK y MK, con $\hat{k} = 5$. GAP y GAPPC encontraron $\hat{k} = 8$.

Como se puede observar, no existe un consenso en el número de clusters con los distintos métodos que hemos utilizado. Sobre todo, se pone de manifiesto que es difícil establecer una partición rígida en datos de expresión génica y en lugar de ello, se encuentran grupos de genes que se comportan de manera similar, aunque distintos métodos pueden diferir en la forma de agrupar tales genes. Em cualquier caso, estos resultados muestran que tanto SK como MK encuentran un número de clusters apropiado y con significado biológico en los datos, aunque

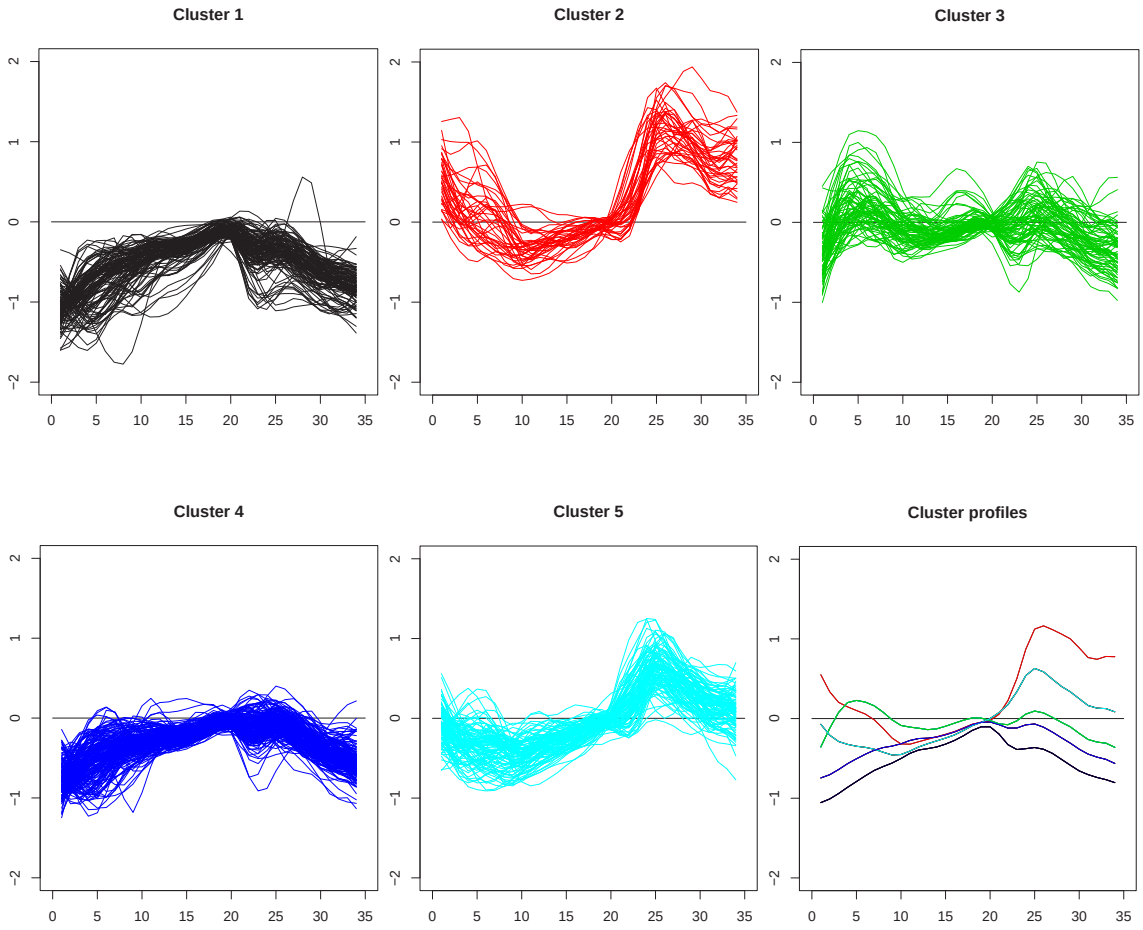


Figura 4.15: Cinco clusters de los datos de S.cerevisiae, obtenidos mediante k-medias. El último gráfico representa los correspondientes perfiles.

TD pierde mucha información, encontrando sólo un grupo. ■

Chapter 5

Conclusions and future work

Summary

We finally summarize the main findings of this work related to different aspects of cluster analysis. Most importantly, we have proposed a technique that allows the comparison between two clusterings, either hierarchical or flat, and its visualization, a method to refine the initial state of k-means, using the multivariate data depth concept and three approaches to estimate the number of clusters underlying in a data set. We also propose some research lines that are derived from this work.

5.1. Conclusions

In this work we have developed several algorithms intended to solve problems related to cluster analysis. The main conclusions are the following:

- The study of the relationships between the clusters from two different clusterings obtained either by the use of different methods or by the use of the same method initialized with different parameters is not always straightforward, and it is useful to analyze many-to-many relationships between clusters on each side rather than one-to-one or one-to-many. The method developed in chapter 2 provides a visualization of the relationships between different clustering results, including hierarchical clusterings without any *a priori* given cut-offs, allowing the user to explore them.

- The use of a scoring function in the case of comparing a flat and a hierarchical clustering provides an analytical manner to decide where to cut the dendrogram, either if it is based on the mutual information between the clusterings or on the graphs layout aesthetics, without selecting *ad hoc* clusters in the hierarchical tree, as it is commonly done.

- In some cases it is possible to improve the original clustering. For instance, in simulated data, if the noise level is such that most of the elements in the clusters are closer to the centers of their 'true' clusters than to the centers of other clusters, our method effectively allows restoring the 'true' clusters. When the noise level increases to the effect that no clustering structure is left, restoring the 'true' clusters becomes difficult. Nevertheless, we have shown that when applied to real gene expression data, we can obtain biologically meaningful results, which helps expert biologists explore their data.

- Our visualization algorithm of the comparison between a hierarchical and a flat clustering effectively defines an order of the hierarchical tree. It is sometimes wrongly assumed by naïve users that a hierarchical clustering has a defined order. In fact,

every branch in the tree can be flipped without affecting the clustering, and in this way, the same clustering can be presented in rather different ways. Applying the cluster comparison method developed in this work reduces this ambiguity.

- The concept of the data depth combined with clusters obtained by k-means is useful to refine the initial state of the algorithm. In particular, alternating the search of the generalized band deepest point in each cluster with the assignment of points to the closest deepest point results into a technique, the GBDRk, that often improves the clustering obtained by k-means, correctly allocating points in their corresponding cluster. The technique is fast and efficient, and it is not sensitive to the presence of noise variables, as it is the case for PAM or CLARA. However, in some cases this technique fails as it is sometimes unable to scape from bad local optima found by k-means.

- The combination of the generalized band depth computation with the bootstrap technique allows the development of a very efficient and robust algorithm, the BRk method, to refine the starting points of k-means, having as well the advantage of being extremely fast compared to DDclust, another data depth-based clustering algorithm.

- The BRk method can be used to design a soft (non-hard) clustering technique, based on the computation of generalized Voronoi regions, that is useful in some contexts such as gene expression data, where a single gene can be related to different sets of genes, and thus be likely to belong to more than one cluster.

- The clustering comparison method defined in chapter 2, following bootstrap results about the distribution of the number of superclusters, can be used to design an algorithm, based on sampling a population of clusterings on the same data set, to estimate the number of groups underlying in the data. We used two different approaches depending on the number of clusters used to generate the population of partitions. The first one, with clusterings with the same number of clusters, is

computationally expensive but provides a nice initial estimation to run the second one, with clusterings with different number of clusters. This second approach is fast and leads to accurate estimations. Both approaches have the drawback of being dependent on the quality of the clustering results, but they achieve good results in simulated and real data.

- The use of the data depth concept combined with the idea of measuring a cluster size through its diameter leads to an alternative to estimate the number of clusters. This approach is very fast and gets good results for a variety of data sets. It fails when the data are not smooth and the clusters have a large overlap when represented in parallel coordinates.

5.2. Future work

The research carried out for this work suggests several topics of future research, described in the following sections.

5.2.1. Comparison of two hierarchical clusterings

Can the method described in section 2.3 be generalized for comparing two hierarchical clusterings? As previously mentioned, there are several important reasons that make this more difficult. First of all, when comparing two hierarchical clusterings there are considerably more degrees of freedom - both trees can be cut at many different levels to obtain flat clusterings. For instance, we have to avoid trivial solutions - perfect correspondence between the roots of the two trees or the leaves of the two trees by adding appropriate terms to the scoring functions. In practical applications the visualization of the correspondence between the clusters plays a central role, and such visualization is more difficult when comparing two hierarchical clusterings.

5.2.2. Comparison of a flat clustering and Gene Ontology

Another complex situation is that of comparing a flat clustering and the hierarchy described by the Gene Ontology (http://www.geneontology.org), an international collaboration among scientists at various biological databases, whose objective is to provide controlled vocabularies for the description of the molecular function, biological process and cellular component of gene products.

The complexity of this comparison arises from the fact that the Gene Ontology hierarchy is not a binary tree, and nodes at higher levels can be descendants of different nodes at lower levels, thus making the cut-offs difficult.

5.2.3. Comparison of a hierarchical clustering to predefined gene sets

Another interesting subject is to perform the comparison of a hierarchical clustering of a data set versus a predefined collection of gene sets, i.e., to compare clustering on different data sets but having a high enough overlap. This way it would be possible to mine and combine information from different data sets what would result into a greater understanding of the nature of the data.

5.2.4. Comparison of soft clusterings

Gasch and Eisen (2002) used a heuristically modified version of fuzzy c-means clustering to identify overlapping clusters of yeast genes based on published gene-expression data following the response of yeast cells to environmental changes. They validated the method by identifying groups of functionally related and coregulated genes, and in the process they uncovered new correlations between yeast genes and between the experimental conditions based on similarities in gene-expression patterns. Futschik and Kasabov (2002) used fuzzy c-means clustering to achieve a robust analysis of gene expression time-series, addressing the issues of parameter selection and cluster validity.

This and other work justify the extension of the comparison methods to the case of having either two soft clusterings or a soft clustering and a standard k-means or hierarchical clustering.

An index to measure overlappings between soft clusters should be defined to take into account the partial memberships of elements.

Apéndice

El algoritmo *correspondence* con exploración ("look-ahead")

Descripción del algoritmo de exploración "look-ahead".

Todas las variables son globales excepto las locales $A, A_1, A_2, A_1^*, A_2^*, \text{parent}, g$ y pointer_2. La configuración inicial es: generation_index = 0, first_split = TRUE, A = root. El usuario establece las posiciones iniciales Y de los clusters no jerárquicos, y el número h de pasos que exploramos.

procedure *look_ahead*(A, h, Y); Y global;
 if A es una hoja
  then
   if A = raiz
    then
     link(A, Y)
    else
     if case $\in \{C, D\}$
      then
       link(A, Y)
  else

encontrar los dos hijos de A : A_1 y A_2

generation_index = generation_index + 1

calcular $\sigma = S(A), \sigma_1 = S'(A_1, A_2)$ y $\sigma_2 = S'(A_2, A_1)$

Y_{12} = gravity_center($(A_1, A_2), Y$)

Y_{21} = gravity_center($(A_2, A_1), Y$)

if $\max(\sigma_1, \sigma_2) < \sigma$

then

if generation_index > h

then

case = A

case_A(A)

else

case = B

case_B(A, A_1, A_2)

else

if first_split = TRUE

then

case = C

case_C(A, A_1, A_2)

else

case = D

case_D()

procedure *case_A*(A)

% no podemos dividir en el h-ésimo paso, por lo que retrocedemos al paso (h-1)-ésimo

link(A, Y)

generation_index = generation_index - 1

procedure *case_B*(A, A_1, A_2)

% no podemos dividir, pero aún podemos explorar algún paso hacia adelante; la primera

% vez que se llama al procedimiento se coloca un puntero global en la rama explorada

if($\sigma_1 > \sigma_2$)

then

A_1^*= A_1

A_2^*= A_2

Y= Y_{12}

else

A_1^*= A_2

A_2^*= A_1

Y = Y_{21}

if first_split = TRUE

then

first_split = FALSE

pointer = A

Y_p = Y

parent = A

descendants = (A_1^*, A_2^*)

if ($\sigma_1 > \sigma_2$)

then

$Y_d = Y_{12}$

else

$Y_d = Y_{21}$

if A_1^* y A_2^* son hojas

then

link(A, Y_p)

else

g = generation_index

link((A_1^*, A_2^*), Y)

look_ahead(A_1^*, Y)

if pointer = parent

then

generation_index = g

look_ahead(A_2^*, Y)

if pointer = parent

then

link(A, Y)

procedure *case_C*(A, A_1, A_2)

% dividimos normalmente

generation_index = 0

if ($\sigma_1 > \sigma_2$)

then

$A_1^* = A_1$

$A_2^* = A_2$

$Y = Y_{12}$

else

$A_1^* = A_2$

$A_2^* = A_1$

$Y = Y_{21}$

pointer = A_1^*

link$((A_1^*, A_2^*), Y)$

look_ahead(A_1^*, Y)

pointer = A_2^*

generation_index = 0

first_split = TRUE

look_ahead(A_2^*, Y)

procedure *case_D*()

% dividimos la rama marcada con el puntero global

$Y = Y_d$

link(descendants, Y)

pointer = descendants(1)

$pointer_2$ = descendants(2)

generation_index = 0

first_split = TRUE

look_ahead(pointer, Y)

generation_index = 0

pointer = $pointer_2$

first_split = TRUE

look_ahead(pointer, Y)

Referencias

Aczél, J. y Daróczy, Z. (1975). *On measures of information and their generalizations.* Academic Press. New York.

Akaike, H. (1974). A new look at statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.

Al-Daoud, M. y Roberts, S. (1994). New methods for the initialization of clusters. *Technical report*, 94.34.

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D. y Levine, A. (1999). Broad patterns of gene expression revealed by clustering analysis of tumour and colon tissues. *Proc. Natl. Acad. Sci.*, 96 (12):6745–6750.

Arabie, P. y Boorman, S. (1973). Multidimensional scaling of measures of distance between partitions. *Journal of Mathematical Psychology*, 10:148–203.

Arcones, M. y Giné, E. (1992). *Bootstrap of M–estimators and related statistical functionals. Exploring the Limits of Bootstrap.* Wiley. New York.

Bach, F. y Jordan, M. (2005). Blind one-microphone speech separation: a spectral learning approach. *Advances in Neural Information Processing Systems*, 17.

Barron, A., Rissanen, J. y Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44 (6):2743–2760.

Bezdek, J. (1981). *Pattern recognition with fuzzy objective function algorithms.* Kluwer Academic Publishers. Norwell, MA, USA.

Bolstad, B., Irizarry, R., Astrand, M. y Speed, T. (2003). A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics*, 19:185–193.

Bradley, P. y Fayyad, U. (1998). Refining initial points for k-means clustering. *Proceedings of the 15th International Conference of Machine Learning.*

Brazma, A., Hingamp, P., Quackenbush, J., Sherlock, G., Spellman, P., Stoeckert, C., Aach, J., Ansorge, W., Ball, C. y Causton, H. (2001). Minimum information about a microarray experiment (miame) –towards standards for microarray data. *Nature Genetics*, 29:365–371.

Breckenridge, J. (1989). Replicating cluster analysis: method, consistency and validity. *Multivariate Behavioral Research*, 24 (2):147–161.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Brown, G. y Cooke, M. (1994). Computational auditory scene analysis. *Computer Speech and Language*, 8:297–333.

Butte, A. y Kohane, I. (2000). Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pacific Symposium on Biocomputing.*

Calinski, R. y Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3:1–27.

Causton, H., Quackenbush, J. y Brazma, A. (2003). *Microarray gene expression data analysis: a beginner's guide.* Blackwell Publishers.

Charikar, M. y Panigrahy, R. (2004). Clustering to minimize the sum of cluster diameters. *Journal of Computer and System Sciences*, 68 (2):417–441.

Chen, D., Toone, W., Mata, J., Lyne, R., Burns, G., Kivinen, K., Brazma, A., Jones, N. y Bahler, J. (2003). Global transcriptional responses of fission yeast to enviromental stress. *Molecular Biology of the Cell*, 14:214–229.

Cheng, C. y Li, L. (2005). Sub-array normalization subject to differentiation. *Nucleic Acids Res.*, 33 (7):5565–5573.

Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D. y Davis, R. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73.

Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. y Futcher, B. (1998). The transcriptional program of sporulation in budding yeast. *Science*, 282:699–705.

Clatworthy, J., Buick, D., Hankins, M., Weinman, J. y Horne, R. (2005). The use and reporting of cluster analysis in health psychology: A review. *British Journal of Health Psychology*, 10:329–358.

de Berg, M., Schwarzkopf, O., van Kreveld, M. y Overmars, M. (2000). *Computational geometry: algorithms and applications.* Springer-Verlag.

DeRisi, J., Iyer, V. y Brown, P. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686.

Doddi, S., Marathe, M., Ravi, S., Taylor, D. y Widmayer, P. (2000). Approximation algorithms for clustering to minimize the sum of diameters. *Nordic Journal of Computing*, 7 (3):185–203.

Donoho, D. y Gasko, M. (1992). Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *Annals of Statistics*, 20:1803–1827.

Dopazo, J. y Carazo, J. (1997). Phylogenetic reconstruction using a growing neural network that adopts the topology of a phylogenetic tree. *Journal of Molecular Evolution*, 44:226–233.

Du, Q., Faber, V. y Gunzburger, M. (1999). Centroidal voronoi tesselations: applications and algorithms. *SIAM Rev.*, 41:637–676.

Dudoit, S. y Fridlyand, J. (2002). A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3 (7):1–21.

Dudoit, S. y Fridlyand, J. (2003). Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19 (9):1090–1099.

Eades, P., McKay, B. y Wormald, N. (1986). On an edge crossing problem.

Eisen, M., Spellman, P., Brown, P. y Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, 95 (25):14863–14868.

Everitt, B. (1980). *Cluster Analysis*. John Wiley and Sons. Halsted Press, New York.

Forbes, A. (1995). Classification-algorithm evaluation – 5 performance-measures based on confusion matrices. *J. Clin. Monit.*, 11:159–164.

Forgy, E. (1965). Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, 21:768.

Fowlkes, E. y Mallows, C. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78:553–584.

Futschik, M. y Kasabov, N. (2002). Fuzzy clustering of gene expression data. *Proceedings of World Congress of Computational Intelligence, WCCI*.

Gansner, E., Koutsofios, E., North, S. y Vo, K. (1993). A technique for drawing directed graphs. *IEEE Trans. on Software Engineering*, 19 (3):214–230.

Garey, M. y Johnson, D. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. W.H.Freeman.

Garey, M. y Johnson, D. (1983). Crossing number is np-complete. *SIAM J.Algebraic Discrete Methods*, 4:312–316.

Garey, M., Johnson, D. y Witsenhausen, H. (1982). The complexity of the generalized lloyd-max problem. *IEEE Transactions on Information Theory*, 28 (2):255–256.

Gasch, A. y Eisen, M. (2002). Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3 (11):1–22.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C. y Lander, E. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537.

Gordon, A. (1999). *Classification*. Chapman and Hall. CRC press, London.

Gross, J. y Yellen, J. (1999). *Graph theory and its applications*. CRC Press.

Grotkjaer, T., Winther, O., Regenberg, B., Nielsen, J. y Hansen, L. (2006). Robust multi-scale clustering of large dna microarray datasets with the consensus algorithm. *Bioinformatics*, 22:58–67.

Gusfield, D. (2002). Partition-distance: a problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82 (3):159–164.

Hansen, M. y Yu, B. (2001). Model selection and the principle of the minimum description length. *Journal of the American Statistical Association*, 96 (454):746–774.

Harper, D. (1999). *Numerical Palaeobiology*. John Wiley and Sons.

Hart, C., Sharenbroich, L., Bornstein, B., Trout, D., King, B., Mjolsness, E. y Wold, B. (2005). A mathematical and computational framework for quantitative comparison and integration of large-scale gene expression data. *Nucleic Acids Res.*, 33 (8):2580–2594.

Hartigan, J. (1975). *Clustering Algorithms*. John Wiley and Sons. New York.

Hartuv, E., Schmitt, A., Lange, J., Meier-Ewert, S., Lehrach, H. y Shamir, R. (1999). An algorithm for clustering cdnas for gene expression analysis using short oligonucleotide fingerprints. *Proceedings 3rd International Symposium on Computational Molecular Biology*.

He, J., Lan, M., Tan, C., Sung, S. y Low, H. (2004). Initialization of cluster refinement algorithms: a review and comparative study. *International Joint Conference on Neural Networks*.

Herrero, J., Valencia, A. y Dopazo, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17:126–136.

Heyer, L., Kruglyak, S. y Yooseph, S. (1999). Exploring expression data: identification and analysis of coexpressed genes. *Genome Research*, 9 (11):1106–1115.

Hoff, K., Culver, T., Keyser, J., Lin, M. y Manocha, D. (1999). Fast computation of generalized voronoi diagrams using graphics hardware. *Computer Graphics*, 33:277–286.

Hubert, L. y Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–198.

Jonnalagadda, S. y Srinivasan, R. (2004). An information theory approach for validating clusters in microarray data. *ISMB*.

Jörnsten, R. (2004). Clustering and classification based on the l_1-data depth. *Journal of Multivariate Analysis*, 90(1):67–89.

Jörnsten, R., Vardi, Y. y Zhang, C. (2002). *A Robust Clustering Method and Visualization Tool Based on Data Depth*. Y.Dodge.

Jörnsten, R. y Yu, B. (2003). Simultaneous gene clustering and subset selection for sample classification via mdl. *Bioinformatics*, 19 (9):1100–1109.

Kapushesky, M., Kemmeren, P., Culhane, A., Durinck, S., Ihmels, J., Korner, C., Kull, M., Torrente, A., Sarkans, U., Vilo, J. y Brazma, A. (2004). Expression profiler: next generation–an online platform for analysis of microarray data. *Nucleic Acids Res.*, 32:W465–W470.

Katsavounidis, I., Kuo, C. y Zhang, Z. (1994). A new initialization technique for generalized lloyd iteration. *IEEE Signal Processing Letters*, 1 (10):144–146.

Kaufman, L. y Rousseeuw, P. (1990). *Finding groups in data: an introduction to cluster analysis*. Wiley. New York.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 9:1464–1479.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1 (1):1–7.

Krzanowski, W. y Lai, Y. (1985). A criterion for determining the number of groups in a data set using sum of squares clustering. *Biometrics*, 44:23–34.

Laszlo, M. y Mukherjee, S. (2006). A genetic algorithm using hyper-quadtrees for low-dimensional k-means clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8.

Leisch, F. (1999). Bagged clustering. *Technical report*, 51.

Li, M. y Vitányi, P. (1997). *An introduction to Kolmogorov complexity and its applications*. Springer-Verlag. New York.

Liu, R. (1990). On a notion of data depth based on random simplices. *Annals of Statistics*, 18:405–414.

López-Pintado, S. y Romo, J. (2006). On the concept of depth for functional data. *Technical report*, 06-30, Statistics and Econometrics Series 12, Department of Statistics, Universidad Carlos III de Madrid.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley simposium in mathematics and probability.*

Meila, M. y Shi, J. (2001). Learning segmentation with random walk. *Neural Information Processing Systems.*

Michaels, G., Carr, D., Askenazi, M., Fuhram, S., Wen, X. y Somoggi, R. (1998). Cluster analysis and data visualization of large-scale gene expression data. *Pacific Symposium on Biocomputing*, 3:42–53.

Milligan, G. y Cooper, M. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

Monma, C. y Suri, S. (1991). Partitioning points and graphs to minimize the maximum or the sum of diameters. *Proceedings of the 6th International Conf. Theory Appl. Graphs*, 2:899–912.

Oja, H. (1983). Descriptive statistics for multivariate distributions. *Statistics and Probability Letters*, 1:327–332.

Peña, D., Rodríguez, J. y Tiao, G. (2004). A general partition cluster algorithm. *COMPSTAT. Proceedings in computational statistics*, 371–380.

Pedrycz, W. (2005). *Knowledge-based clustering. From data to information granules.* Wiley.

Pentney, W. y Meila, M. (2005). Spectral clustering of biological sequence data. *Proceedings of the 25th Anual Conference of AAAI.*

Pollard, D. (1982). A central limit theorem for k-means clustering. *Annals of Probability*, 10:919–926.

Quackenbush, J. (2001). Computational analysis of microarray data. *Nature Reviews Genetics*, 2:418–427.

Radonjic, M., Andrau, J., Lijnzaad, P., Kemmeren, P., Kockelkorn, T., van Leenen, D., van Berkum, N. y Holstege, F. (2005). Genome-wide analyses reveal rna polymerase ii located upstream of genes poised for rapid response upon s. cerevisiae stationary phase exit. *Molecular Cell*, 18:171–183.

Rand, W. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850.

Ripley, B. y Hjort, N. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rissanen, J. (1978). Modelling by shortest data description. *Automatica*, 14:465–471.

Sanguinetti, G., Laidler, J. y Lawrence, N. (2005). Automatic determination of the number of clusters using spectral algorithms. *IEEE Workshop on Machine Learning for Signal Processing*, 55–60.

Sedgewick, R. (1986). A new upper bound for shell sort. *J.Algorithms*, 7:159–173.

Shannon, C. (1948). A mathematical theory of communication. 27:379–423, 623–656.

Sharan, R. y Shamir, R. (2000). Click: A clustering algorithm with applications to gene expression data. *ISMB*.

Shi, J. y Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22.

Shokoufandeh, A., Mancoridis, S., Denton, T. y Maycock, M. (2005). Spectral and metaheuristic algorithms for software clustering. *Journal of Systems and Software*, 77:213–223.

Shokoufandeh, A., Mancoridis, S. y Maycock, M. (2002). Applying spectral methods to software clustering. *Proceedings of the Ninth Working Conference on Reverse Engineereing (WCRE02)*.

Somolonoff, R. (1964). A formal theory of inductive inference. *Information and Control*, 7:1–22,224–254.

Sugar, C. (1998). Techniques for clustering and classification with applications to medical problems. *PhD Thesis*.

Sugar, C. y James, G. (2003). Finding the number of clusters in a data set: an information theoretic approach. *Journal of the American Statistical Association*, 98:750–763.

Swets, J. (1988). Measuring the accuracy of diagnostic systems. *Science*, 240:1285–1293.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. y Golub, T. (1999). Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoetic differentiation. *Proc. Natl. Acad. Sci.*, 96:2907–2912.

Tavazoie, S., Hughes, D., Campbell, M., Cho, R. y Church, G. (1999). Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285.

Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D. y Brown, P. (1999). Clustering methods for the analysis of dna microarray data. *Technical report*.

Tibshirani, R., Walther, G. y Hastie, T. (2001). Estimating the number of data clusters via the gap statistic. *J.R.Statist.Soc.B*, 63:411–423.

Törönen, P., Kolehmainen, M., Wong, G. y Castrén, E. (1999). Analysis of gene expression data using self-organizing maps. *FEBS Lett*, 451:142–146.

Tou, J. y González, R. (1974). *Pattern Recognition Principles*. Addison–Wesley.

Tukey, J. (1975). Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematicians*.

Vardi, Y. y Zhang, C. (2000). The multivariate l_1-median and associated data depth. *Proc. Natl. Acad. Sci.*, 97:1423–1426.

Verma, D. y Meila, M. (University of Washington, 2003). A comparison of spectral clustering algorithms. *Technical report*.

Vilo, J., Brazma, A., Jonassen, I., Robinson, A. y Ukkonen, E. (2000). Mining for putative regulatory elements in the yeast genome using gene expression data. *ISMB*.

Weber, A. (1929). *Theory of location of industries [English translation by C.J.Friedrich from Weber's 1909 book]*. Univ. Chicago Press.

Wen, X., Fuhrman, S., Michaels, G., Carr, D., Smith, S., Barker, J. y Somogyi, R. (1998). Large-scale temporal gene expression mapping of central nervous system development. *Proc. Natl. Acad. Sci.*, 95:334–339.

Yeh, A. y Singh, K. (1997). Balanced confidence regions based on Tukey's depth and the bootstrap. *J.R.Statist.Soc.B*, 59:639–652.